A Decentralised Informatics, Optimisation and Control Framework for Evolving Demand Response Services

by

Sean Williams

Director of Studies: Prof. Michael Short

Supervisor: Dr. Tracey Crosbie

School of Computing, Engineering & Digital Technologies

Teesside University

This thesis is submitted in partial fulfilment of the requirements of the Teesside University for the degree of Doctor of Philosophy

January 2021

Abstract

Centralised energy generation and distribution networks are becoming more vulnerable to energy security. Closure of fossil-fuelled power plants and an increase in more volatile decentralised renewable electricity generation is aggravating the situation further. Future storage technologies will inevitably play a more dominant role during the energy transition. Paradoxically, as the number of renewables increase, there is a greater reliance on conventional power sources in providing back-up supply. Demand response is an important instrument offering a wide range of services how customers can modify their energy consumption when system reliability is jeopardised. This research focuses on integrated demand response in an energy system by evolving a decentralised informatics, optimisation and control framework. The contributions of this research are (1) the development of a low-cost, standalone frequency measurement instrument, (2) a short-term electricity demand forecasting methodology, and (3) an optimisation policy that guides the decision-making process by balancing the building occupant's comfort, cost (tariff) and the current and predicted states of the system. Computer simulation and hardware-in-the-loop testing is used to evaluate an energy system operation. There are three significant findings in this research. First, a prototype frequency measurement instrument output is shown to be as effective as measured grid data. Second, a electricity demand forecaster is likely to have a positive influence on the operation and planning of supply and demand management. Third, the proposed optimisation and control framework reveals the effectiveness of the new methods in tackling the energy optimisation problem. This research recommends deployment of the optimisation and control framework, at scale, as part of a wider integrated demand response scheme for decentralised energy systems.

Publications and Research Outputs

Publications

S. Williams, M. Short, and T. Crosbie, "On the use of thermal inertia in building stock to leverage decentralised demand side frequency regulation services," *Applied Thermal Engineering*, vol. 133, pp. 97–106, 2018.

*This paper forms the basis of **Chapter 3**. It examines the effects of grid frequency and thermal mass in buildings using a prototype frequency instrument. The pro-active frequency control regulation method intends to provide a fast-acting balancing mechanism to support energy management.*

S. Williams, and M. Short, "Electricity demand forecasting for decentralised energy management" *Energy and Built Environment*, vol. 1(2), pp. 178-186, 2020.

*In this work, we formulate a simple yet highly effective method for forecasting the rate at which electricity is consumed. **Chapter 4** focuses on the implementation of the method described in this paper.*

S. Williams, M. Short, T. Crosbie, and M. Shadman-Pajouh, "A Decentralised Informatics, Optimisation and Control Framework for Evolving Demand Response Services" *Energies*, vol. 13(16), 4191, 2020.

*A closed-loop optimisation and control framework for energy management presented in this paper provide the foundation of **Chapter 5**. Early deployment activities, including details of experimental testing, are then communicated in **Chapter 6**.*

Conference Papers

S. Williams, M. Short, and T. Crosbie, "Evaluating the role of building thermal inertia for the provision of decentralised demand-side primary electrical frequency regulation services" in the 4th Sustainable Thermal Energy Management International Conference, 2017.

This preliminary paper introduced the findings of the prototype frequency instrument design and initial test results. The work progressed, reaching a standard acceptable for publication in Applied Thermal Engineering.

S. Williams, and M. Short, "Decentralised energy optimisation for blocks of buildings" in International Conference on Innovative Applied Energy, 2019.

The proposed energy optimisation solution conceptual framework was first introduced to the community at this international conference. The idea of formulating an optimisation algorithm based on a weighted directed graph was widely accepted. This work was subsequently extended before it was accepted for publication in the acclaimed Energy and Built Environment journal.

S. Williams, and M. Short, "Electricity demand forecasting in decentralised demand side response for blocks of buildings" in International Conference on Energy and Sustainable Futures, 2019.

This Doctoral Training Alliance hosted event provided an opportunity to present the early findings of a series of computer simulations designed to functionally test the optimisation algorithm. The positive feedback helped guide the research towards a credible energy management framework and later selected for publication in the Energies journal.

Acknowledgements

I want to express thanks to Teesside University and the Doctoral Training Alliance scheme in Energy for their support, which I have enjoyed by attending various energy events and sharing ideas with its members.

I want to give sincere respect to my supervisory team Prof. Michael Short and Dr. Tracey Crosbie, together they have guided and encouraged new ways of thinking. They have always been willing and enthusiastic to assist, providing insight and expertise that helped the research.

I wish to acknowledge the support and great love of my wife, Debbie, especially during the compilation of this thesis.

"It is not the mountain we conquer, but ourselves."

Sir Edmund Hillary

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This submission is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Sean Williams

January 2021

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

Nomenclature

Symbols

$\triangleright$	algorithm comment
$a_i, \ldots, d_i$	cubic polynomial coefficients
c	weekly seasonality period index
cf	correction factor
D	damping constant
d	day
df	degrees of freedom
dfv	demand forecast value
DIR	detect increase (or decrease)
dt	data type; $dt \in \{\texttt{comfort,demand,tou,optim}\}$
ET_{th}	energy tariff threshold
FIT	energy storage asset availability
H	inertia time constant
Hf	high frequency
h	hour
hi	end data point of PAA segment
I	PI controller integral gain
k	number of equal-sized segments
Ki	ALFC secondary loop gain
l	lower input range (min-max feature scaling)
Lf	low frequency
lo	start data point of PAA segment
min	minute
mo	month
n	count of measure of an entire population or sample
n	number of observations
P	PI controller proportional gain
Pd	nominal demand load

PWR	power source
R	DFC-Primary regulator
s	standard deviation
$S0$_date	stage 0 date time group
sec	second
$S_i(x)$	piecewise function
σ_n^2	population variance
S_n	stage Sn; $n \in \{0, 1, \ldots, 23\}$
s_n^2	sample variance
SOC_{des}^{th}	state of charge demand event threshold
SOC_{max}^{th}	state of charge maximum threshold
SOC_{min}^{th}	state of charge minimum threshold
t	time
T_g	governor time constant
T_h	heat pump thermal time constant
T_{min}^{th}	minimum temperature threshold (°C); $T_{min}^{th} \in \{15.5, \ldots, 17.5\}$
t_mode	tariff mode
tou	time of use
T_{room}	room temperature (°C)
T_{S1}	temperature setpoint (°C); $T_{S1} \in \{15.5, \ldots, 20.5\}$
T_{step}	temperature step increase (°C); $T_{step} \in \{2, 3\}$
u	maximum of input range (min-max feature scaling)
wk	week
$\boldsymbol{x_i}$	k-dimensional vector of $\overline{x}_i$
$\overline{x}_i$	mean value of each segment
x_{max}	maximum value of x in range
x_{min}	minimum value of x in range
y_i	original value of y at time t
yr	year
y_t	value of y at time t

Acronyms / Abbreviations

ACF	Autocorrelation Function
ACO	Ant Colony Optimisation
ALFC	Automatic Load Frequency Control
ANN	Artificial Neural Networks
AOC	Automatic Optimisation & Control
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
BEM	Building Energy Management
BESS	Battery Energy Storage System
BMRS	Balancing Mechanism Reporting Service
BP	Back Propagation
CBR	Case Base Reasoning
Cmean	Cumulative Mean Value
CNN	Convolutional Neural Networks
DFC	Decentralised Frequency Control
DIN	Deutsches Institut für Normung
DNO	Distribution Network Operator
DP	Dynamic Programming
DR	Demand Response
DREG	Distributed Renewable Energy Generators
DSO	District System Operators
DSR	Demand Side Response
DWT	Discrete Wavelet Transform
ESM	Exponential Smoothing Methods
ESO	Electricity System Operator
ETS	Error, Trend and Season (Exponential Smoothing Methods)
F	Functional
FOPDT	First Order Plus Dead Time
FTS	Fuzzy Time Series
GA	Genetic Algorithm

GDP	Gross Domestic Product
GHG	Greenhouse Gas
HIL	Hardware-in-the-loop
HVAC	Heating, Ventilating and Air Conditioning
IDE	Integrated Development Environment
IEA	International Energy Agency
IMC	Internal Model Control
LEM	Local Energy Management
LS	Least Squares
LSSVM	Least Squares Support Vector Machines
LSTM	Long Short-Term Memory
LUT	Lookup Table
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MARS	Multivariate Adaptive Regression Spline
MCU	Microcontroller Unit
MILP	Mixed Integer Linear Programming
ML	Machine Learning
MTWFT	Days of week
NF	Non-Functional
N	New
OLS	Ordinary Least Squares
PAA	Piecewise Aggregate Approximation
PPD	Predicted Percentage of Dissatisfaction
PSF	Pattern Sequence Forecasting
PSO	Particle Swarm Optimisation
PMV	Predicted Mean Value
RMSE	Root Mean Square Error
ROCOF	Rate of Change of Frequency
SARIMA	Seasonal Autoregressive Integrated Moving Average
SAX	Symbolic Aggregate Approximation
SDR	Self-Discharge Rate

SD	Secure Digital
SOC	State of Charge
SPM	Sequential Pattern Mining
SSSP	Single Source Shortest Path
SS	Days of weekend
SLTF	Short Term Load Forecast
SVD	Singular Value Decomposition
SVM	Support Vector Machines
SVR	Support Vector Regression
TCL	Thermostatically Controlled Load
TOU	Time Of Use tariff
TSO	Transmission System Operator
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UTC	Coordinated Universal Time
U	Updated
VAR	Vector Regression
WI	Willmott's Index
WMT	Weighted Matrix Table

Chapter 1

Introduction to Research Project

1.1 Problem formulation

This research deals with decentralised closed-loop control and optimisation for energy (electrical) management.

Until recently, the UK electricity grid was predominately supplied by synchronous fossil-fuelled power plants connected at the transmission level in a centralised network. Now, an increasing number of distributed generation resources (e.g., from solar and wind) operating on an interconnected system, has helped steer the energy sector on a pathway towards a low carbon future [1]. However, closure of larger traditional fossil-fuelled power plants, driven mainly by environmental considerations, advances in technology and geopolitical influence, has highlighted one of the most predominant technical challenges faced by system operators. Non-synchronous machines are generating an increasing amount of power at the distribution level, which is reducing system reserve capacity and making power grids less resilient to frequency imbalances [2, 3].

In conventional power stations, generators provide inertia as they rotate at the same frequency of the electrical grid. The inertia acts as a short-term buffer against sudden change. However, the reduced level of inertia provided by synchronous generation is not always sufficient to maintain system frequency within acceptable operating standards. Therefore, during periods when generation output from renewable sources ceases, it is more challenging to keep the frequency within its standard operating range. A decentralised approach offers greater regional flexibility when balancing supply and demand.

The proposed decentralised, informatics, optimisation and control framework for energy management has access to different resources that can be made available depending on the urgency and scale of the imbalance. Integrating elements of smart-grid technologies and

improved coordination between energy communities and distribution network operator has the potential to unlock new flexible, decentralised control measures and encourage more active customer participation in demand side response (DSR) programmes [4].

A recent study concluded that a decentralised approach to energy curtailment by exploiting thermal inertia in building stock is possible when participating in pro-active demand response using frequency regulation [5]. A key consideration when taking part in a pre-defined energy reduction strategy must empower customers to use energy in the lowest price period accessible, at the same time as offering participation in DSR events. This research provides a novel perspective by placing the building and its occupants as an integral part of a much more inclusive energy management system.

This research is centralised around a demand response strategy which proposes to decompose the overall approach into two main parts. The first uses grid frequency to moderate space heating in a building. Arresting grid frequency excursions, through load shifting of heating and cooling units in real-time can have a positive influence on reserve generation capacity without compromising user comfort. The second part requires knowledge of future electrical demand, tariff, and user feedback on perceived thermal comfort. A resultant energy optimisation and control scheme offers primary and secondary demand response for energy management.

This chapter presents an overview of the research study. Aims and objectives of the research are listed before the methodologies specific to this work are introduced. Details of a computer simulation for energy management (including its development) indicate the scale of work. Accordingly, a set of design constraints and limitations are given before the contribution to knowledge is stated. Finally, this chapter summarises the thesis organisation. Figure 1.1 shows a graphical abstract of the research development.

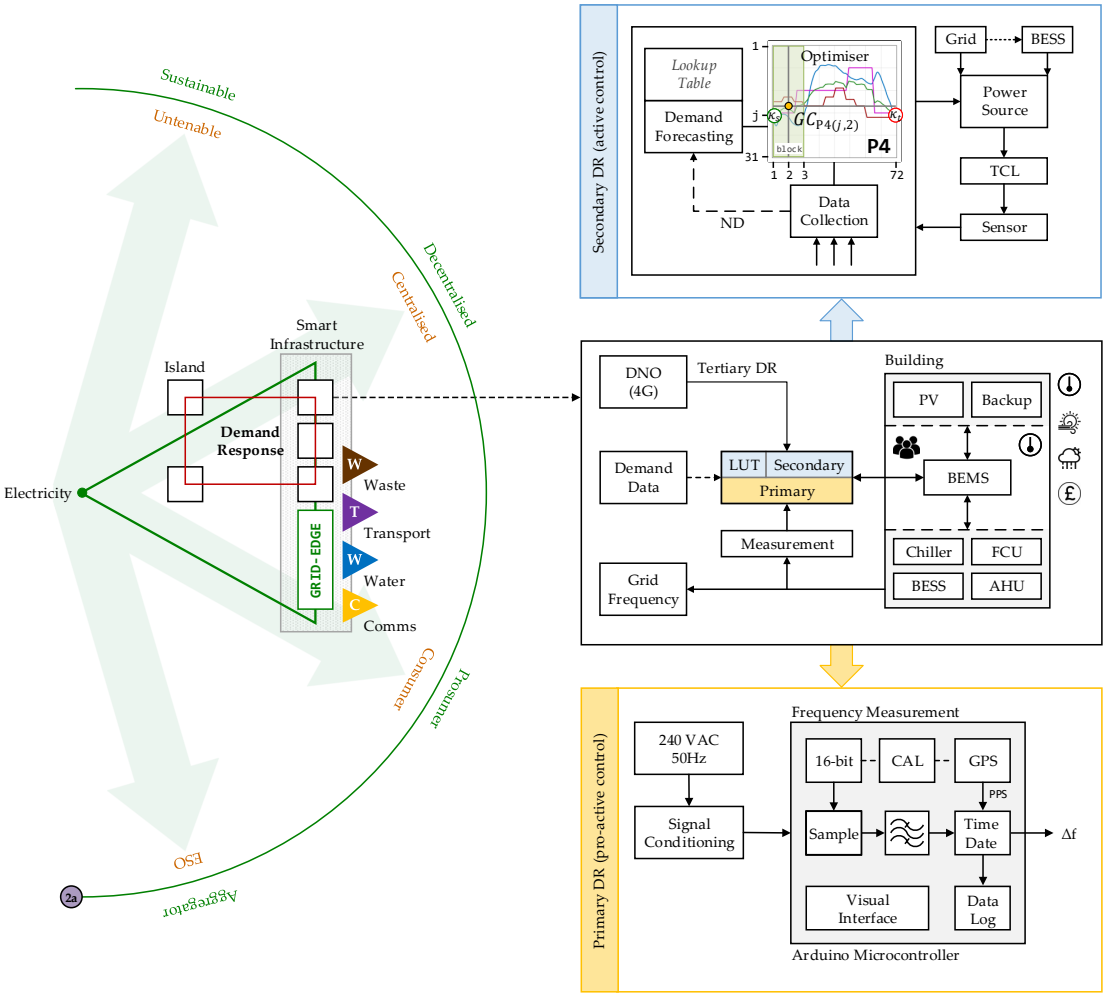


Figure 1.1 Research overview

1.2 Research aim

Real-time control and optimisation will play a vital role in future power grids. Decentralised energy networks and community-driven energy management schemes hold high potentials in terms of local grid stabilisation and for sustainable energy.

The research aim is to advance integrated demand response in a decentralised energy system.

1.3 Objectives

The following objectives form a pathway to fulfilling the declared research aim:

1. Develop a low-cost, standalone grid frequency measuring instrument capable of working in conjunction with an optimisation and control scheme designed to moderate space heating in a building based on local needs.
2. Formulate an energy forecast algorithm by analysing demand (electrical) time series data.
3. Implement integrated demand response in an energy system capable of automatically arresting the severity of supply-demand imbalances.
4. Use prototype hardware to evaluate a demand response approach with real-world data.

The findings could be useful to operators of community power systems that aspire to power a sustainable future. Here, providing decision-making tools to assist in short to medium term energy planning or as part of an evolving integrated demand response service. Areas where generation capacity margins are narrowing, or a need to improve the resilience of power generation systems to help improve the stability of the network.

1.4 Methodologies

This section presents each of the research methodologies identified to achieve the research study objectives.

Step 1: Literature review

A critical review aims to highlight the significance and originality of the research presented. At the same time, as justifying work packages listed, a literature review seeks to summarise the current state-of-the-art in energy management systems for decentralised or small remote energy communities. The literature review dedicates one section to each of the following topics: (1) electricity supply and demand, (2) energy consumption, and (3) optimisation scenarios. The focus will be on identifying existing methodologies and expose gaps in

knowledge that justify the deployment of decentralised automatic control and optimisation methods for energy management.

Step 2: Understanding the relationship between grid frequency and thermostatically controlled loads

As a demand response resource, thermostatically controlled loads can respond to power fluctuations caused by intermittent distributed renewable energy generators on the network. When used to provide space heating in buildings, the slow varying thermal inertia means occupants can remain satisfied with their thermal environment during these short-term transient excursions.

Decentralised demand side frequency regulation when used in building stock can regulate short-term frequency excursions in demanded electrical energy. The proposed decentralised demand response method can operate with no national communication infrastructure but requires access to a reliable source of grid frequency. Hence, a low-cost microcontroller platform capable of monitoring and recording grid frequency at the point of connection is developed. By connecting the device to the grid frequency, a series of hardware-in-the-loop real-time simulation tests, it is possible to assess the overall impact on thermal comfort due to fluctuations in measured grid frequency.

Step 3: Understanding energy (electrical) consumption

The aim is to understand daily electricity consumption and develop an algorithm that will estimate future demand. A forecasting session is constructed initially through analysis of a chronological sequence of discrete observations. Then, using dimensionality reduction techniques and piecewise interpolation, an electricity demand forecasting method is created. Providing energy consumption information can inform the proposed energy management optimiser.

Step 4: Understanding near real-time closed-loop control and optimisation for energy management

Complex problems are solved using the proposed optimisation algorithm. The optimality depends on the problem and the algorithm used to achieve the best performance. This

research presents an optimisation method that was inspired by Bellman's Principle of Optimality [6]. The optimisation process for energy management proposes to influence space temperature setpoint using a weight-based routing algorithm.

Here, the demand (electrical consumption), tariff and building occupants satisfaction rating of the thermal environment are tested, and sequence of future actions obtained to optimise energy consumption and automatically schedule use of energy storage assets. A few assumptions were made during the development of the optimiser.

Despite the high rate of energy consumption by heating systems in buildings, evidence suggests occupants are not always satisfied with their thermal environment. Therefore, this research proposes a framework that considers individual thermal comfort satisfaction. The method reacts to data collected using smartphone technology. The algorithm learns aggregated thermal comfort preference profiles, which informs the optimiser. Software development and hardware-in-the-loop testing validate the smartphone application functional requirements. Then, computer-based simulation aggregates the single-use application to replicate multi-user environments.

Step 5: Understanding the resulting implications

Inferential statistical analysis is used during quantitative analysis that sought to collect, analyse, and interpret grid frequency data. Next, a comprehensive series of computer-based simulation and experimental tests are undertaken using a prototype platform to test and evaluate the optimisation and control algorithm in real-time.

1.5 Key contributions of the research

The key contributions of this research are:

- Combining ideas to design and test a low-cost, standalone frequency measurement instrument to assess the relationship between grid frequency and load disturbance. *Publication: Applied Thermal Engineering, vol. 133, pp. 97-106, 2018.*

- Developing a new mathematical model to calculate the rate of energy (electrical) consumption. *Publication: Energy and Built Environment, vol. 1(2), pp. 178-186, 2020.*
- Re-contextualisation of an existing technique, by applying a weight-based routing algorithm in a new context, using a computer-based simulation to demonstrate integrated demand response in an optimisation and control framework for energy systems. *Publication: Energies, vol. 13(16), 4191, 2020.*

1.6 Scope and limitations

The grid frequency transmitted from a power station in the UK is nominally 50 Hz. Control activities used during this research are tailored around this value. Therefore, deploying the software to areas in the world that uses a utility frequency other than 50 Hz will first require an update.

By design, the optimisation algorithm temperature range works between 15.5 °C and 20.5 °C. A software update is required to operate outside these parameters.

1.7 Thesis organisation

Figure 1.2 shows an overview of the thesis organisation comprising seven chapters and related appendices. A summary of each chapter is set out below.

Chapter 1 summarises of the research project. Aims and objectives are listed, and methodologies introduced. Key contributions of the research are given together with notes that describe known limitations.

Chapter 2 begins with a critical review of energy management systems. The chapter reviews the development and application of different approaches developed to improve the efficiency of energy use, which includes demand response services. The evolution of closed-loop control and optimisation techniques used to bring benefit are also discussed.

Chapter 3 follows the development of a grid frequency measurement instrument and provides information on the proposed prototype hardware platform and software environment. An insight into univariate time series forecasting that comprises grid frequency is briefly discussed. The chapter concludes by presenting test results that validate the prototype frequency measurement tool.

Chapter 4 focuses on implementing the electricity demand forecasting method. An electricity demand forecasting method is constructed initially through analysis of a chronological sequence of discrete observations. A series of simple mathematical transformations complete the process. Test results of the forecasting method are presented.

Chapter 5 first introduces the closed-loop optimisation and control scheme, which is an important contribution to the demand response strategy. Next, a technical description of a series of simulation models designed to test a decentralised community energy management system is presented. The simulation comprises a simplified lumped model for electrical demand forecasting introduced in Chapter 4, a scheduling subsystem that optimises the utility of energy storage assets, and an active/pro-active control subsystem. A multi-objective cost function provides secondary demand response services formulated using a weight-based routing algorithm. Results of a series of simulation tests are presented.

Chapter 6 performs early deployment activities using prototype hardware in an experiment designed to test the interaction of energy assets for optimal near real-time closed-loop control with real-world data. Special attention is given to the control actions that underpin the effectiveness of the proposed demand response strategy. The chapter begins with a review of hardware selected to complete experimental testing. Details of a smartphone app designed to allow building occupants to report relative thermal comfort levels are then presented. Details of hardware configuration and set-up of the experimental environment are described before test results are presented.

Chapter 7 summarises significant findings and sets out recommendations for future work.

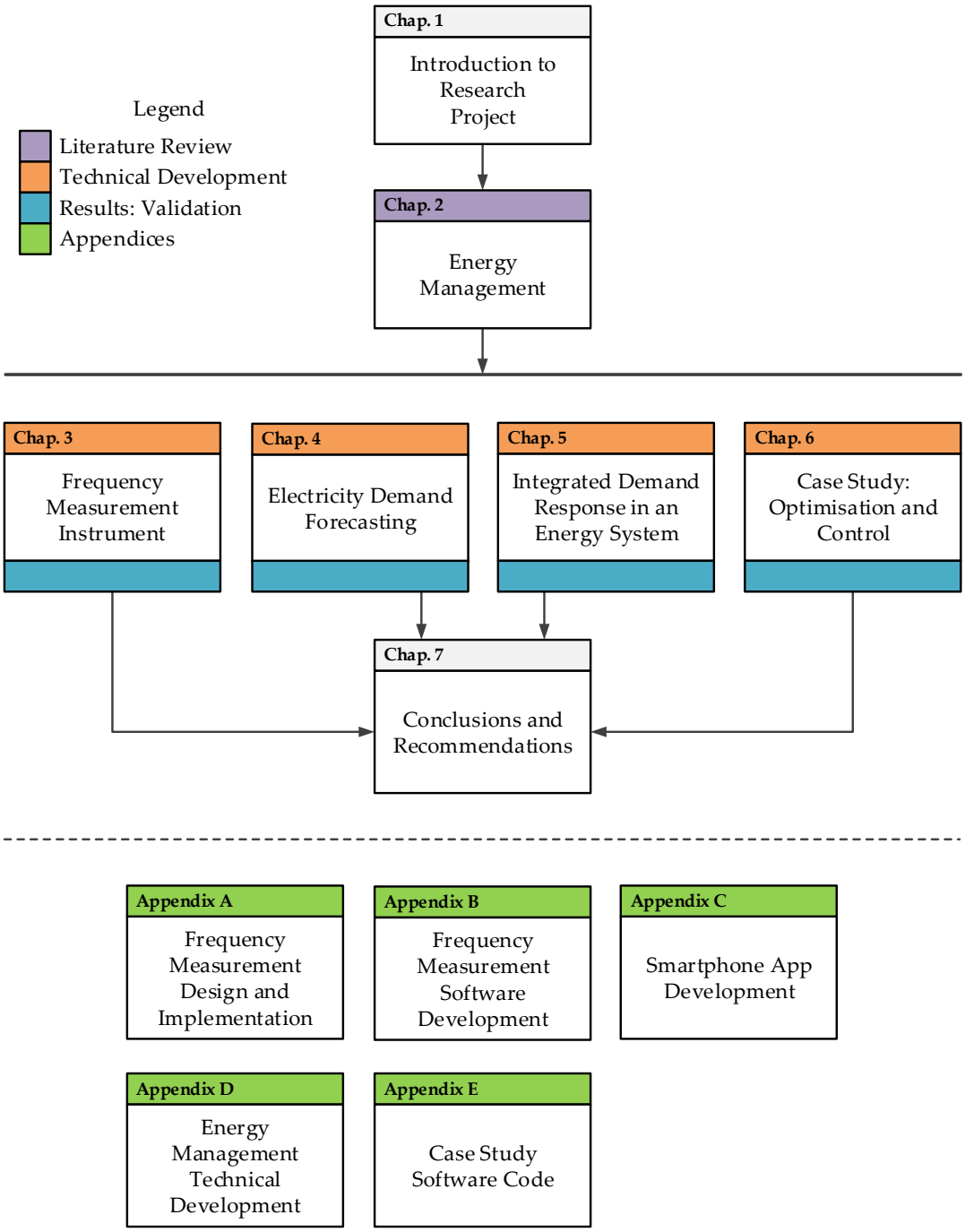


Figure 1.2 Thesis organisation

1.8 Thesis Y-shaped matrix diagram

A matrix diagram that links the different elements of the thesis is shown in Figure 1.3. The diagram intends to help identify relationships between objectives, chapters (including

appendices), journal publications and methodologies discussed earlier. Links are graded primary, secondary, and minor.

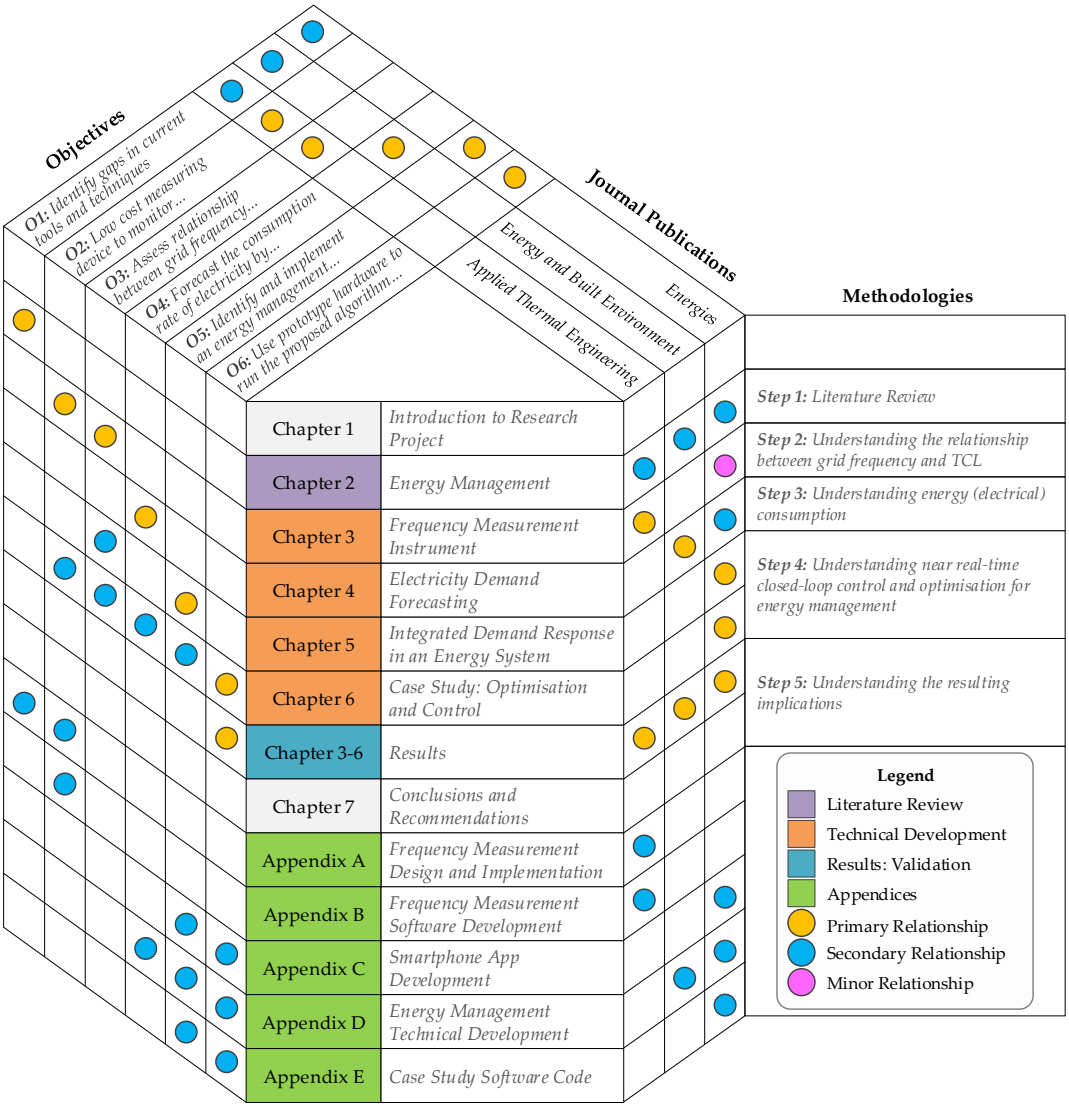


Figure 1.3 Y-shaped matrix diagram

Chapter 2

Energy Management

2.1 Introduction

Nowadays, the challenges and opportunities which are related to energy systems are diverse and complex. The energy transition is usually defined as a structural change that aims to bridge the energy divide by delivering low carbon and net zero solutions. Technology innovations are helping to reduce environmental stress, at the same time as providing greater flexibility and energy equality. The pressures to mitigate climate change are driven by new knowledge and growing expectations in society, which are later translated into national and regional policy change. It is precisely this perspective of diverse complexity that is reflected in the energy trilemma (see, e.g., [7]). The energy trilemma conceptual framework aims to balance energy security, energy equality and environmental sustainability [8]. Navigating the energy transition successfully promotes policy coherence and greater cooperation at the highest levels, which implies effective management and potential trade-offs. At a national level, steering towards a sustainable energy future means managing electricity supply and demand effectively.

Therefore, this chapter begins by introducing the concept of electricity supply and demand. Then, it examines energy consumption analysis before reviewing different energy optimisation scenarios.

2.2 Electricity supply and demand

The energy system can be categorised into two divisions, which are usually known as the supply side and demand side. Traditionally, power generation, conversion, storage, transmission, and distribution reside in the supply side, whereas the consumers of energy

reside in the demand side. The supply output supports the highest peak load, which is because of coincidental usage, driven by end-user groups. To maintain the balance between load and generation, there is a need to adjust the generating capacity of centralised power plants constantly. Demand response services can provide benefits when system operators find it increasingly challenging to align generation and end user demands.

The effectiveness of modern technologies continues to improve energy efficiency. However, this does not translate to a fall in energy demand [9]. Reduction in energy consumption due to technology improvements, somewhat paradoxically, causes energy actors to consume more energy [10]. There is evidence that ongoing trends in energy consumption exist on both the production (supply) and consumption (demand) side [11]. While policy interventions are advancing technology and economic growth, this strong coupling is causing environmental stress [12]. Therefore, it is essential to improve energy access that is sustainable to help mitigate risks associated with one of the most extraordinary growth paths in modern times. The ever-increasing presence of sustainable energy supply is lessening harmful emissions from fossil fuel power plants, which contribute to a rise in greenhouse gases [13].

Nevertheless, the intensified uncertainties associated with modern power systems operating close to their stability boundaries, means system network operators are facing acute challenges when maintaining continuity of supply [14]. Demand response is an essential tool in the energy systems of many developed and industrialised countries. In a future power system, where the contribution of inertia alone can no longer provide resilience during sudden changes in frequency, demand response provides an effective mechanism to help balance supply and demand [15].

Traditionally, electricity markets have evolved on the assumption that electric utilities and system network operators will supply all power demands whenever they occur [16]. However, centralised generation and distribution through an ageing infrastructure of high voltage distribution networks to regional system operators are becoming more vulnerable to energy security [17, 18]. In 2015, circa 80% of global energy consumption was generated using fossil fuel [19]. Delivery of low carbon, energy-efficient solutions have become more prevalent in recent years [20]. The move away from large fossil fuel power plants operating on a centralised configuration is motivated by greater digitisation, the drive for decarbonisation

and a need for more customer control in energy management [21]. Therefore, to achieve carbon reduction goals, an obvious decarbonisation strategy is to extend fuel mix diversity in the electricity sector while displacing the highest polluting power plants [22].

The accelerated transformations in our energy system are not unique. Similar complex changes are becoming more noticeable across all sectors in society. The scale and fusion of technologies are impacting how governments manage the economy, how businesses react to profound technological innovation and affecting how people live. The so-called Fourth Industrial Revolution is creating new technologies that form an industry of networks, platforms, and digital innovations. Technological breakthroughs in areas from big data to information intelligence promote the efficiency of resources, reduces costs, and improves the quality of human life [23].

In energy systems, the progress of renewable sources, the innovation of distribution grids, and investment in energy storage solutions are attributed to Industry 4.0 developments. These evolving power solutions and the emergence of the prosumer are shaping the energy markets for innovation [24]. The drive towards a smart and flexible energy system is a crucial element of modern industrial strategy. Moreover, the integration of industrial development and alignment of environmental goals have been advocated by several recent reports [25, 26]. Likewise, Lütkenhorst et al. [27] identified that policymakers need to create incentives in a coordinated way to ensure progress on all fronts simultaneously. If governments invest in low carbon technologies or other intermittent sources of power, they must also ensure simultaneous investments in smart grids and energy storage solutions to ensure grid stability.

In the UK, the number of decentralised energy operations is on the increase [28]. These changes are motivated in part by an increasing political drive in response to environmental policy priorities. Consequently, this is provoking a shift towards decentralised energy systems and business models that involve community energy groups simultaneously [29]. Innovations in energy evolution are characterised in part by industrial strategy and relations to decarbonisation [14]. The fall in the cost of renewables has been significant in the last 10 years, which means generating electrical energy from renewables is cheaper [30]. Nevertheless, when combined with an increased burden on present-day centralised services, risks associated with long-term supply security and the drive to be carbon neutral by 2050 are exposed. While

market signals and shifts in government policy are guiding the energy sector transformation, system operators have developed many control strategies to preserve equilibrium in grid frequency during periods of peak demand, including demand response.

The UK government has set ambitious targets for electric cars and electrification of heating [31]. These bold steps are accelerating the decarbonisation of vehicles and encouraging innovation in electrification technologies, which will further increase the demand for electrical power. The recent emergence of smart cities and communities helps population clusters to become more efficient and, their energy infrastructures more sustainable [32, 33]. By integrating smart technologies, coupled with a network of sensors and intelligent algorithms, it is often reported that urban smartness is at the forefront of the sustainability transition [34]. However, while the development of smart grids is necessary to modernise the electricity market, many of the reported environmental and security benefits are realised only when smart technologies are combined with decentralised energy generation [35]. In sustainable development scenarios, a transition towards low carbon energy will operate on different geographical scales. Increased customer participation and increased demand require the decentralisation of energy supply [36]. Smart (energy) cities should not only support local needs in terms of energy demands but also feature broader regional or national network demands. However, while the development of smart grids is necessary to modernise the electricity market, it is only when combined with decentralised energy generation, many of the reported environmental and security benefits are realised [37]. Besides this, demand for new building stock continues to accelerate, driven in part by renewed industrialisation and economic growth [38].

Studies have highlighted that building energy consumption and contribution to greenhouse gases is significant [39]. In smart energy developments, regulatory control of heating, ventilating and air conditioning (HVAC) processes in buildings, and other thermostatically controlled loads, make them exceptionally suitable candidates for providing energy flexibility to the grid [40]. Many control strategies that aim to improve the operation of heating systems have been proposed (e.g., see [41–43]). The slow thermal dynamics and rather stochastic characteristics of buildings (including occupants) mean their power consumption can be easily shifted as part of a demand response mechanism without causing a significant short-term impact on space temperatures in controlled environments [44].

Developing energy efficiency in energy systems is perhaps the most sustainable way to help reduce carbon emissions [45]. Providing access to electricity brings many socio-economic benefits [46]. Electricity consumption, particularly by industrial and commercial sectors and services, is the resource that allows other services (such as education, health, drinking water and sanitation) to be provided [47]. Various studies have shown how small-scale distributed renewables are changing people's lives. However, many island energy communities fall behind mainland energy network developments when it comes to securing affordable and sustainable supplies. Community energy networks that operate a small number of DREG are often more exposed to system vulnerabilities because of their intermittent nature of energy production [48]. Still, for population clusters dependent on conventional diesel generators, decentralised developments offer an alternative sustainable clean energy transition pathway. More recent studies show that low carbon smart energy systems offer interconnected islands new opportunities for energy independence [49]. Harvesting energy from natural resources to achieve specific targets of decarbonisation can be realised using smart energy systems combined with efficient control strategies aimed at balancing the energy demand and production [50, 51].

The energy market is moving from a linear centralised system to a more flexible, complex, and decentralised system. A decentralised approach can deliver electricity in a controlled environment providing network operators access to frequency regulation and balancing services [52–54]. Flexibility in energy generation and utility becomes prevalent in small geographical areas. A smart grid approach provides technology infrastructure opportunities that enable intermittent DREG to connect with local battery energy storage systems. However, it is important to note that distributed energy installations require coordination mechanisms, especially when network operators request flexibility in consumer behaviour to secure operation of the power system.

In small island communities, optimisation and control of decentralised energy systems may bring economic reward, improve energy security, and offer new opportunities for consumers to become more active in energy management [55]. Even so, one of the main challenges integrating several intermittent DREG is the power systems ability to respond to a change in demand. In the absence of robust communication networks or problems due to latency, the ability to respond quickly enough is often problematic [56]. In contrast, local direct

control demand response processes may offer a more reactive approach by redistributing energy consumption in response to changes in grid frequency measured at source. However, motivations for decentralisation are not universally consistent, and embracing a carbon reduction pathway through decarbonisation initiatives is not always the main priority for instigating change [57]. Therefore, these schemes must not be to the detriment of the end customers, such as adversely affecting the thermal comfort of building occupants or loss of essential services [58]. It is important to note that substituting energy from fossil fuels with suitable sustainable energy sources to meet the needs and expectations of the community, will help improve the quality of human life [59].

The achievement of a decentralised energy system requires the integration of multiple natural resources, often supplemented by some form of reserve capacity (e.g., electricity storage systems for providing ancillary services or diesel generators for back-up power). If the benefits of low carbon power systems within a decentralised setting are to be achieved, then energy management mechanisms must be capable of coordinating and managing a flexible set of services, each characterised by local resources [60]. Alongside the physical transformations, demand side management becomes the most critical dimension, especially when there is a tendency to empower consumers to generate their electricity [61]. A recent study highlights that prosumers are likely to play a crucial and enabling role in a decentralised system [62]. Ultimately, efficiency improvements established using optimisation and control algorithms (demand side management) will help lower emissions and supply needs.

As a general proposition, the objective for energy planning is to develop a system that satisfies a dynamic energy forecasting need for community needs and is consistent with sustainable development scenarios. In contrast, the objective of the optimisation procedure will be formulated during the analysis of energy potentials and their geographical location. Such expositions suggest optimisation problems may be categorised as either one-dimensional or multi-dimensional depending on the pre-defined objectives [63]. However, with energy efficiency, there is ample evidence that shows that most optimisation problems are defined by at least two objectives: time and energy. In practice, many real-world problems are defined as a process of finding a minimal value of an n-dimensional function subject to a set of constraints that may or may not be related [64].

The control of power demand in response to variations in grid frequency is an essential part of the smart grid vision. In demand response, existing research methods can be broadly divided into two types, where one method focuses on classical demand response programmes such as direct control, and initiatives that aim to curtail energy consumption during peak times, usually through financial incentives. Furthermore, robust communication protocols are needed to supervise interaction between network operators. Islands have often served as test platforms for distributed smart energy systems [65]. However, for most remote communities that do not attract the same level of energy technology innovation, such an architecture is out of reach.

The emergence of community energy has attracted much attention in academia in recent years [66–68]. The idea that the three predominant categories of energy actors (i.e., community, state, and private sector) are separated is challenged in a recent study [69]. It is argued, community energy requires the active participation and support of all actors [70]. There are many examples in literature that recognise the importance of community engagement [66, 71]. Entanglement of actors and technologies shows there is a growing need for a new role that supports energy transition at a community level [72]. Doing so will help establish shared visions at the same time as forming support networks that shape future technology innovations.

2.3 Energy consumption analysis

Recent trends toward decarbonisation, digitalisation and decentralisation are seeking to build out centralised state-owned power assets, focusing instead on investments in energy storage, demand response and improved energy efficiency [36]. In the energy field, these principles are often expressed in smart infrastructure initiatives that are focused on increasing the capacity of low carbon technologies while improving the efficiency and resilience of energy production [73]. The pattern of growth and expansion of the world human population is set to increase annually, rising to about 11 billion in 2100 [74]. In other studies, it is shown human population density peaks in high productivity environments [75]. An analysis of the casual relationship between the economy and energy showed that electricity consumption is a critical component of economic growth [76]. The literature on electricity consumption and

economic growth relations highlights four conflicting hypotheses, including feedback effect, growth, conservation, and neutral effect [77].

These findings have significant implications for nations planning for an energy transition. For countries confirming the feedback hypothesis, the implication is that economic growth and electricity consumption are mutually dependent [78]. In such a case, policymakers should concentrate on electricity generation policies and economic growth policies that stimulate each other. However, while shared goals and objectives might motivate a common political agenda, different policy design for more developed countries should probably be considered. The data in previous studies that examined the Granger causality relationship (see, e.g., [79]) between energy consumption and real gross domestic product, show energy conservation policies, if implemented correctly, will have no adverse impact on economic growth in more developed countries [80]. In other studies which analysed countries homogeneous concerning their level of development, no casual relations in the group of wealthiest countries were observed [81]. In contrast, evidence of energy-growth nexus in groups of developing countries can be found [82].

Constructing energy systems into more sustainable forms means electricity demand forecasting is necessary. As a broad guideline, research has shown that energy consumption in buildings accounts for approximately 40% of the world's energy resources and emits circa one-third of greenhouse gases [83, 84]. Considering the long lifespans and complex challenges associated with the regeneration of old building stock, more accessible energy retrofit initiatives to achieve energy saving targets are needed [85]. Tangible measures that improve energy efficiency include lifestyle changes, e.g., use of smart meters [86], and distribution system planning as well as enhancing load and resource forecasting methods and approaches [87].

Time series data analysis is found in many sectors including financial [88], transport [89], retail [90] and health care [91]. The aim is often focused on identifying underlying components in data (deterministic and stochastic) including trend, seasonal, cyclical, and calendar variations. Analysis usually means describing them mathematically and making predictions or forecasts about what will happen next. Decisions formulated on empirical analysis can help in effective decisions regarding rail transport planning and management (transport), provide a basis for distribution and replenishment plans (retail), or allow for a more reliable approach

to intensive care therapy (health care). In energy consumption analysis, a time series of demand data can be defined as a set of chronologically ordered points observed over time and subsequently used for time-based predictions. Knowledge about future electricity demand ensures supply and demand management decisions help balance the electricity generation and usage [92].

Many technical barriers make forecasting of electricity demand challenging, especially in areas that support a combination of different distributed renewable energy generators that lack the flexibility and capacity offered by centralised energy systems. Analysis of temporal data and the development of forecasting models are often presented as multivariate time series problems [93–96]. However, multivariate time series considers simultaneous time-dependent variables where each variable depends not only on its past values but also has some dependency on other variables. Thus, a multivariate prediction may prove difficult to extract enough meaningful information useful for predicting future states. In contrast, a univariate time series with a single time-dependent variable may offer an improved alternative when prediction time horizons are small [97].

Research investigating temporal data and the prediction of future values in time series highlights there is little consensus around the terminology that defines the duration of each forecasting horizon. However, most time series forecasting problems in literature can be framed as short-term, medium-term, or long-term, depending on the domain and the underlying process. For instance, in many economic applications, weekly, monthly, quarterly, and annual trends are clear. However, estimating annual or quarterly seasonal adjustments when the number of recent observations used in the estimation is limited to the previous 12 months, will prove problematic. Similarly, one year of daily activity would not estimate annual seasonality accurately. Therefore, the number of observations used in the estimation (referred to as the window size) is a crucial issue in forecasting [98]. For stationary and ergodic processes, a forecast content function is formulated to determine a forecast horizon beyond which forecasts continue to convey useful information from univariate time series models [99]. The results provide a characterisation of the conditioning information at different horizons, which serve as a useful benchmark. It is worthy of mentioning, while traditional time series methods (e.g., autoregressive integrated moving average (ARIMA), seasonal ARIMA (SARIMA) and error, trend and season (ETS)) handle single seasonality in a time

series, more advanced techniques may be required when multiple seasonality components exist in data [100].

Energy demand models can be classified in several ways, such as static versus dynamic, univariate versus multivariate and techniques ranging from vanilla method approaches to hybrid models. A considerable amount of literature has been published on energy consumption prediction methods, including conventional statistical-based methods, classification-based, support vector machines (SVM) and artificial neural network (ANN) methods.

ARIMA models and exponential smoothing are amongst the most general form of time series forecasting techniques. ARIMA models are based on the idea of transforming the time series to be stationary by first applying differencing operations. In this context, a stationary time series is when the statistical mean, variance and autocorrelation are all constant over time. In contrast, exponential smoothing is a time series method for univariate data. Unlike the ARIMA model where the prediction is a weighted linear sum of recent past observations (or lags), the exponential smoothing method uses an exponentially decreasing weight of past observations. There are three main types of exponential smoothing forecasting methods used in time series. The more advanced method, known as the Holt-Winters exponential smoothing method, adds support for seasonality to univariate time series [101].

Ediger et al. [102] presented a method to estimate future primary energy demand of Turkey from 2005 to 2020 using the ARIMA and SARIMA methods. The method integrates each model by using specific decision parameters related to goodness-of-fit and confidence interval, the behaviour of the curve, and reserves. The results show that the ARIMA forecasting of the total primary energy demand appears to be more reliable than the summation of individual forecasts.

Noureen et al. [103] emphasised the need to apply a differencing operation to non-stationary process before applying an ARIMA model for forecasting seasonal agricultural loads. The ARIMA model is based on the behaviour of observed data and completely ignores the independent variable. The results are reported as competitive; however, comparing results from different models would benefit the study.

Taylor [104] considered five exponentially weighted methods when formulating forecasting up to one day ahead using half-hourly load data. An empirical comparison of univariate methods tested the forecasting accuracy. The results showed a new singular value decomposition (SVD) based exponential smoothing formulation outperformed all other methods on load forecasting applications. The SVD enables a multivariate dataset to be reduced to a dataset of lower dimension.

Arsenault et al. [105] predicted the total energy demand as a function of the previous year's energy demand, price of energy, real income, and heating day for the province of Quebec. The ordinary least squares technique (OLS) is used, and prediction is made sector-wise, i.e., residential, commercial, industrial, and street lighting. Yearly data has been used for demand side projection. Weather data influences energy forecasts.

Machine learning (ML) techniques have recently been proven to be workable and effective in analysing time series data [106–109]. In the field of deep learning (a subset ML which deals with neural networks), long short-term memory (LSTM) can be applied to time series forecasting. Somu et al. [110] developed an energy consumption forecasting model which uses LSTM and improved sine cosine optimisation algorithm for accurate and robust building energy consumption forecasting. Experiments reveal that the proposed model outperforms the state-of-the-art energy consumption forecast models in terms of mean absolute error, mean absolute percentage error, mean square error, and root mean square error.

Yang et al. [111] recognised the importance of optimal feature selection. The proposed hybrid model that combines least squares support vector machines (LSSVM) and autocorrelation function (ACF) selects the optimal input features and predicts half-hourly electricity loads of the following week. When compared with other benchmark models (Bmean, Bplag, Bpday and Bpweek), experimental tests provide more accurate half-hour ahead short-term load forecast (STLF). Despite this, the proposed hybrid model is very time consuming, and the algorithm is complicated.

Sadaei et al. [112] proposed a multivariate short-term load forecasting method combining fuzzy time series (FTS) and convolutional neural network (CNN), a class of ML that have been shown to provide state-of-the-art results on recognition tasks. This novel hybrid approach to convert multivariate time series into images and then using FTS and CNN

provided good results for STLF when compared to other benchmark models, including LSTM.

Al-Musaylh et al. [113] conducted a study that focused on data-driven techniques for forecasting short-term demand data using several forecast horizons. A single demand data was used to develop the univariate ARIMA model. When compared to multivariate adaptive regression spline (MARS) and support vector regression (SVR) methods, normalised model assessment metrics based on root mean square error (RMSE), mean absolute error (MAE) and Wilmott's Index (WI) (see, [114]), show MARS and SVR models are more suitable for STLF in Queensland, Australia.

Bio-inspired meta-heuristic optimisation algorithms, which can solve difficult optimisation problems, have gained popularity in the past decade. Some of the latest techniques, such as Bayesian vector autoregression (VAR), ant colony optimisation (ACO), particle swarm optimisation (PSO) models, are being used in energy demand analysis. VAR models are well-liked for their flexibility and rich parameterisation. In recent years, Bayesian VAR forecasting models have demonstrated considerable success in forecasting macroeconomic and regional economic variables. Despite this success, these promising forecasting models have yet to be widely used in energy forecasting. However, drawing on Bayesian VAR econometric modelling techniques, Njenga et al. [115] developed a mortality model that allows qualification of parameter uncertainty in a prediction distribution. Comparisons with other models using univariate techniques show that Bayesian VAR can improve mortality model fits.

Toksari [116] utilised historical data between 1970 and 2005 to train an ACO electricity energy estimation model to estimate the electrical energy demand in Turkey in the years 2006 to 2025. The models which are obtained using ACO included four economic parameters, including population, gross domestic population, import and export. The findings proved the ACO approach to be a successful energy estimation tool.

Ozerdem et al. [117] modelled the problem of STLF using a proposed optimisation of designed feedforward neural networks using the PSO algorithm. The system was trained using a backpropagation (BP) neural network, and the learning rate adjusted until an optimal result for the network was established. The results obtained within this work showed that both

particle swarm optimised neural network and BP neural network are suitable for modelling load forecasting. It is also observed that the required times for training the BP networks are roughly twice of the particle swarm optimised networks. Therefore, faster models can be developed with PSO networks.

In a further example that utilises the PSO network, El-Telbany et al. [118] present a method developed to forecast the Jordanian electricity demand. Results are compared with outputs from the BP algorithm and autoregressive moving average methods. The PSO intelligent based load forecasting technique performed better than the BP algorithm. However, the PSO requires many more function evaluations to find the optimal solution, as compared to BP.

Time series forecasting algorithms can create models based on historical observations to estimate future behaviour. Recently, the concept of 'big data' has occupied academia and business. Buildings have not only become more energy-intensive, in the era of smart technologies, they have also become more data-intensive. The computation time of more traditional time series methods may increase notably when big data time series is tested. Techniques more familiar in data mining can play an important role in big data time series. According to Sumathi et al. [119], data mining can detect and extract hidden relationships, patterns, and trends by search through extensive data. Alvarez et al. [120] presented a new approach called pattern sequence forecasting (PSF) to forecast energy time series. The process utilises the k-means clustering technique to reduce the dimensionality of the database. Clustering generates a sequence of labels which define a pattern of search, and finally, the prediction step is defined. This novel approach avoids the use of real values of the time series until the last step of the prediction process. The algorithm has been successfully applied in electricity price and demand time series of Spanish, Australian, and New York markets.

Manojlović et al. [121] proposed a novel time series grouping algorithm that combines dimensionality reduction, both partitional and hierarchical clustering, and cluster validation to group time series into an optimal number of clusters based on simple parametric settings. Case study results on real smart meter data confirm the proposed algorithm achieves high cluster validity.

It is well known that the dimensionality curse destructively impacts on the result of time series data mining. That is, as the number of features grows large, poor generalisation is to be expected and, training becomes intractable due to high computational and memory costs (see, e.g., [122]). The most prominent methods used to alleviate the dimensionality curse include discrete wavelet transform, piecewise aggregated approximation (PAA) (see, [123]) and symbolic aggregated approximation (SAX) (see, [124]). The latter being an extension to the PAA, which transforms the mean values derived from PAA into discrete string symbols.

Wang et al. [125] extend the idea of PAA by using vector quantised approximation. This approach allows for a more flexible approximation of each segment, which is represented by a codeword derived from a codebook of key sequences. Tests using real and simulated datasets show that the proposed technique generally outperforms traditional PAA methods where the size of each segment is constant.

2.4 Optimisation scenarios

The design of many engineering solutions often involves many complex processes. A design often follows an iterative and incremental life cycle, where each end of cycle design is based on experience, intuition, and perhaps mathematical reasoning. The objective is identifying a design that is optimal according to a specified statistical optimality criterion. An optimal design process forces the design engineer to identify a set of design parameters, an objective function, and any constraints the design must operate [126]. This approach allows engineers to model systems and make these trade-offs, one of which can integrate regulatory requirements, or guidelines to make solutions more acceptable to society.

In energy systems, the field of optimisation has received much attention in recent years. Advances in computing power, availability of user-friendly software solutions and new approaches to optimisation, has expanded our knowledge in this area of research. For example, in 1995, inspired by swarm intelligence of fish and birds, and even by human behaviour, Kennedy et al. [127] developed a modern meta-heuristic algorithm. The diversity of these nature-inspired algorithms has become increasingly popular, solving hard optimisation problems in all major branches of science and engineering. When applying machine learning,

swarm intelligence has been widely applied to feature selection because of its simplicity, effective search mechanism and natural representation [128].

In statistical analysis, the methods of least squares (LS) try to minimise the sum of residuals, e.g., see [129]. However, obtaining the LS estimate is cumbersome when the number of measurements is large. More recently, ML has been used to learn patterns from data and then make decisions to optimise the deviation between what is observed in data and what the model predicts. It should be noted that ML does not necessarily guarantee improvements in performance [130]. When ML was applied to improve time series forecasting accuracy, the findings of a study showed pure ML methods performed poorly when compared to statistical methods [131].

Most control and optimisation theories have been developed out of *a priori* reasoning based on relatively simple assumptions that the reader can distinguish 'optimal control'. Optimisation is a key idea that can be applied in many situations. However, any mathematical interpretation is ultimately a crude (or sophisticated) simplification of any real-world application. In practice, the optimisation criteria will largely be determined by interpreting the mathematical model and its associated problem description (e.g., variables, objective function, and constraints). It is convenient to classify each type of problem as stochastic or deterministic [132]. The first frames the optimisation problem in the presence of uncertainty, whereas in contrast, it is appropriate to consider the second optimisation problem as a sequence of operations over time, each leading to a uniquely determined state. For such an algorithm, there is some trade-off between accepting the most optimal outcome (cost) at the present stage against the least total cost incurred from all subsequent stages. Genetic algorithms (GA) and hill-climbing with random restart serve as good examples of stochastic algorithms [133].

GA is a random search method and can act directly on the offspring of optimisation results. It is often used in discrete optimisation and life prediction of big data. The leading operators in GA are cross over, mutation, and selection of the fittest. Liu et al. [134] used GA to search the optimal value in an online energy management system based on driving status recognition. When compared to rule based controls, which are not adapt to complex energy systems, GA methods are regarded as feasible alternative.

The inherent intermittent behaviour of renewable energies means representing uncertainties in optimisation problems becomes critical. Interestingly, stochastic optimisation in renewable energy applications has been shown to deliver accurate representations in capturing the uncertainties of renewable systems [135]. In a grid that comprises solar and wind generators, Altintaş et al. [136] state deterministic approaches alone cannot capture the dynamics of the system. To overcome this, a multistage stochastic programming model was formulated to handle the uncertainties related to renewable generators and evolving economic environment, [137–139].

Generally, most existing commercial building energy management systems (BEMS) adopt demand-driven control strategies [140]. In the last three decades, ANNs have been utilised to solve several architectural and civil engineering problems (see, e.g., [141, 142]). However, considering the application of the ANN in buildings, Mohandes et al. [143] reviewed the potential of ANN for prediction of buildings energy consumption. According to studies performed by Olofsson et al. [144], ANN has a superior performance compared to the other methods for estimating energy consumption in buildings. Macarulla et al. [145] proposed a control system based on a neural network that determined the optimum time to turn on a boiler to achieve comfort levels. Results showed that implementing predictive control in a BEMS for building boilers can reduce the energy required to heat the building without compromising the user's comfort. In another study, Yuan et al. [146] combine LSTM and PSO to optimise parameters for improved prediction interval of wind power. Unlike the ANN, LSTM (a form of recurrent neural network) shares parameters across different time steps because of a recurrent connection on the hidden state. Many contributions in literature combine forecasting and optimisation methods (e.g., see, [147, 148]).

One drawback of BEMS, when deployed to implement demand side management is the lack of integration of energy consumption data into actionable information [149]. Pombeiro et al. [150] compared the performance of two optimisation models to control space heating subject to economic and thermal constraints. Using a simplified thermal model and step time of 10 min, test results showed that the DP optimisation approach performs better than the GA.

Niu et al. [151] investigated the potential of exploiting building thermal energy and battery storage. The results highlighted that a linear autoregressive model with exogenous inputs

could accurately predict thermal load with a 60 min horizon. Besides, a mixed-integer linear programming (MILP) model was formulated for optimal dispatch of a building energy system. The results show that the operational costs decreased when using battery energy storage, and further cost savings are achieved when using building thermal inertia.

Work presented by Short et al. [152] builds on recent MILP models. The short-term forecasting, along with scheduling and economic dispatch optimisation for small/medium scaled decentralised combined heat and power plant, was formulated. The results show that profit is much more sensitive to the accuracy of load predictions when compared to previous studies in this area.

Recent innovations of case-based reasoning (CBR) model are well documented (see, e.g., [153]). Faia et al. [154] proposed CBR approach, uses previous cases of energy reduction in buildings to suggest ideal levels of energy reduction to be applied in the energy consumption of houses. Using k-means clustering to search for clusters of similar past cases, the optimisation process, PSO was utilised to optimise the choice of the variable that characterise each case. The results show that the combined CBR and PSO approach can identify adequate levels of energy reduction without compromising the thermal comfort of building occupants. However, the performance of CBR model is dependent on the number of similar cases. Another limitation of this model is that a small deviation from the original optimisation problem requires a significant change.

In a different study, Delgarm et al. [155] formulated a multi-objective optimisation method for building energy efficiency by integrating an artificial bee colony algorithm with a building energy simulation tool. Unlike other population-based meta-heuristic models, this global based approach preformed well noting that lower energy consumption led to increasing predicted percentage dissatisfaction (PPD) values.

Ghahramani et al. [156] developed a HVAC system energy optimisation using an adaptive hybrid meta-heuristic. By using a combination of k-nearest neighbour stochastic hill-climbing, regression decision tree and a recursive algorithm, the knowledge-based approach was used for determining the initial temperature setpoint. The energy savings analysis presented did not factor thermal comfort constraints. Besides this, results for different permutations of the algorithm showed an overall improved energy consumption.

In many real applications the problem description cannot be adequately represented as a single objective function. Other studies have also concluded that a multi-objective optimised design is better than two individual single objective optimised designs [157]. In the context of multicriteria optimisation, Granat et al. [158] formulated a Pareto-optimal algorithm (see, [159]) where it became necessary to choose a reasonable solution rather than the best for all criteria. The contribution of the solution process guides the operator to select a Pareto-optimal path that best fits their preferences using an interactive graphical representation of a grid network. There are similarities in this approach and the optimisation problem formulated using dynamic programming. However, like most multicriteria shortest path problems, if the polynomial complexity is left unbound the computational effort required to solve such problems increases exponentially with the size of the problem.

Kapsalis et al. [160] presented an optimal operation scheduling algorithm based on Dijkstra's algorithm. The algorithm was designed to control thermostatically controlled devices (electric water heaters) continuously. Here, user preferences formed part of the multi-objective function, which influenced the edge weight between successive nodes. The static single source shortest path algorithm calculates the optimal path for different edge weights. This iterative process achieves the minimisation of energy costs while ensuring the user thermal preferences are not jeopardised. In this scenario, the performance was satisfactory, based on the polynomial complexity was bounded.

Minimising or maximising an objective function helps solve an optimisation problem. Often, optimum scenarios consist of a mixture of technologies and range of operating modes. In recent years demand response has become more prominent, especially with the advances in artificial learning [161]. A critical review of energy system models suggest they can be separated into four main groups. Advances in technologies have seen the emergence of more complex approaches. However, while the decisive advantage of artificial intelligence is its ability to work with noisy or incomplete data, there are distinct disadvantages when compared to more traditional optimisation approaches. Table 2.1 sets out the main advantages and disadvantages of the four main groups of energy system models, which includes the dynamic programming approach. Generally, dynamic programming is a method which can efficiently deal with linear and non-linear objectives and constraints and output satisfactory optimal solutions.

The composition of the energy problem set out in this study is based on the adopted optimisation criterion involving dynamic programming, which are mainly classified into problems with discrete and continuous spaces. Mavrovouniotis et. al [162] identified most practical real-world problems consist of a finite number of solutions, including problems with network environments. Therefore they can be formulated as discrete optimisation problems. The dynamic programming method is mainly used to deal with energy management and optimal control problems of hybrid energy plants [163]. The characteristics are seen as most suitable given the overall problem solution can easily be translated into a number of smaller decisions, following the principle of optimality. Also, given the approach simplistic design and implementation, its diffusion of its use for energy systems and asset scheduling applications operating as part of a wider integrated demand response framework, ensures an optimal solution is always found.

Table 2.1
Optimisation methods: advantages and disadvantages

Method	Advantages	Disadvantages
Linear Programming/MILP	1. Simple to implement 2. Can be used across multiple domains It can manage multiple factors to solve complex problems, which leads to a better quality of solution 3. Accurate	1. Only applicable for linear problems 2. Long computational times 3. Approach relies on linear equations 4. Efficiencies of scale often do not relate to linear effects 5. The problem must be converted into a mathematical model to use Linear Programming 6. Assumptions made in linear programming are unrealistic, because a linear relationship assumes that factors never change
Swarm Intelligence	1. Simple concept and extendable to different domains 2. Easy implementation 3. Robustness to control parameters 4. Comutational efficiency when compared to other mathematical and other heuristic optimisation techniques	1. Lacks solid mathematical foundation for analysis 2. Has some limitations for real-time economic dispatch applications since the PSO is also a variant of stochastic optimisation techniques requiring longer computation time than mathematical approaches 3. Current PSO algorithms require a range of parameters to be tuned for different problems and applications 4. Generalised parameter values and learning components are required so the approach can be used in different problem domains

continued . . .

. . . continued

Method	Advantages	Disadvantages
Generic Algorithm	1. Optimises both linear and non-linear problems 2. Is faster and more efficient when compared to traditional methods 3. Can be used for discrete and continuous problems	1. Difficult to define initial parameters 2. Time consuming for complex problems 3. Optimal solution may not possible 4. Not suitable for all problems 5. Maybe computationally expensive
Dynamic Programming	1. It is applicable to a wide range of problems, including linear, non-linear,deterministic or stochastic 2. Can be used for discrete and continuous problems 3. Suitable for complex systems such as chemical engineering design and energy management 4. Global optimum is assured 5. Can be applied to any problem that observes the *principle of optimality*	1. Difficult to define initial parameters 2. Decomposition of problem and storing intermediate results can be memory intensive 3. Each problem has to be modeled according to its unique constraints and specification

2.5 Summary

Traditionally, a centralised power system is scheduled to generate electrical energy because most loads are not measurable at the required time resolution. Maintaining frequency equilibrium is challenging, exacerbated further by a growing number of diverse renewable power generation operating on the grid. The characteristics of renewable energy often mean their operation is geographically dispersed, decentralised. Demand response has emerged as one of the most important instruments to reduce electricity during critical peak periods or shift the demand to off-peak periods. However, managing this flexibility in future energy systems requires expansion scenarios that recognise the energy autonomy aspirations at the community level.

Due to the variability of renewable resources, constructing energy systems into more sustainable forms means electricity demand forecasting is necessary. Studies in the literature consider a range of methods using different forecast horizons. Often, the simplicity of conventional approaches outperform more advanced modern techniques. A critical review of optimisation methods has shown these can be separated into four main groups. The advantages and disadvantages of each group shows not one method fits all optimisation problems. Despite advances in technologies, and increased attention on the development of energy systems, the advantages set out at Table 2.1 warrant further work to resolve an energy optimisation problem based on dynamic programming. Also, prior studies that provide an integrated demand response for community energy systems have not been found provides further justification for pursuing this area of research. Specifically, this is no evidence that brings the collective contributions of energy forecasting, active and pro-active demand response, respecting occupant thermal comfort preferences at the same time as considering the economic impact as part of a multi-objective optimisation problem. A gap in knowledge has been identified.

Chapter 3

Frequency Measurement Instrument

3.1 Introduction

There is a tangible link between economic growth and increased demand in energy for space heating and air conditioning [164]. Therefore, thermal storage in buildings as a resource is of growing interest. According to recent studies, energy consumed in residential and commercial buildings accounts for circa 20% of the globally delivered energy [165]. Building efficiency strategies can be: (1) passive, i.e., seeking to improve the fabric of buildings, and (2) active, i.e., encompassing improvements to space heating by decreasing the energy demand of the building [166, 167]. Balancing services by manipulating the load profile of domestic appliances or exploiting the amount of thermal mass in a building has received some attention [168, 169]. However, the literature on the provision of similar gains through decentralised pro-active frequency regulation and optimisation of indoor comfort temperatures in commercial building stock by exploiting properties that contribute to thermal stability is less apparent [170].

Differences between previous approaches to DR and the one presented in this chapter are twofold. Firstly the presented method is not dependent upon a national ICT infrastructure (decentralised control). Secondly, the approach avoids discontinuous (on/off) switching of loads, thus, avoiding synchronisation and restoration loading issues. The method discussed here sets out to replicate the primary 'droop' control present on supply generators on the demand side, exploiting electro-thermal couplings. It can operate independently at the building or block of buildings scale, in either islanded mode or on-grid. Also, it controls space heating, without compromising occupancy thermal comfort by using a signal set to change proportionally to grid frequency. Hence, avoiding long-term oscillatory behaviour [171, 172]. The findings demonstrate that this research offers a simple and viable alternative to other market mechanisms, such as fast-acting frequency response services.

Provision of primary frequency control has been discussed extensively in the literature [173, 111]. However, the drawback, when operating in the context of demand response is its dependency on a robust communication system [174]. Aggregation of TCLs presents unique challenges as a reactive control strategy attempts to minimise the synchronisation phenomenon, usually by dispatching some form of elaborate stochastic switching mechanism. Concerning existing literature, this work proposes a pro-active decentralised approach to demand response while preserving the characteristics necessary for primary frequency control.

The main contribution of this work is the development of a prototype low-cost standalone grid frequency instrument that can be deployed without the reliance on centralised communication for accurate local frequency measurement. A review of measured frequency and data obtained from transmission system operators makes a comparison between different solutions and methodologies more credible. Providing a near real-time grid frequency measurement designed to operate and output accurate frequency measurements within the locality of the device is key to supporting a primary control mechanism for pro-active energy management.

3.2 Materials and methods

3.2.1 Methodology

Decentralised demand side frequency control when used in building stock can regulate short-term frequency excursions in demand electrical energy [175]. The proposed decentralised demand response method can operate with no national communications network but requires access to a reliable source of grid frequency measurement. Figure 3.1 illustrates the general approach. Here, when a power disturbance (ΔPd) is applied to a single area power system driven by a lumped parameter non-reheat steam turbine, the decentralised frequency control action can have a positive influence on reserve generation capacity. A change in reserve generation capacity is achieved by arresting the measured frequency excursion in real-time. Moreover, a small variance in building (zonal) temperature setpoint will have a negligible effect on occupant thermal comfort.

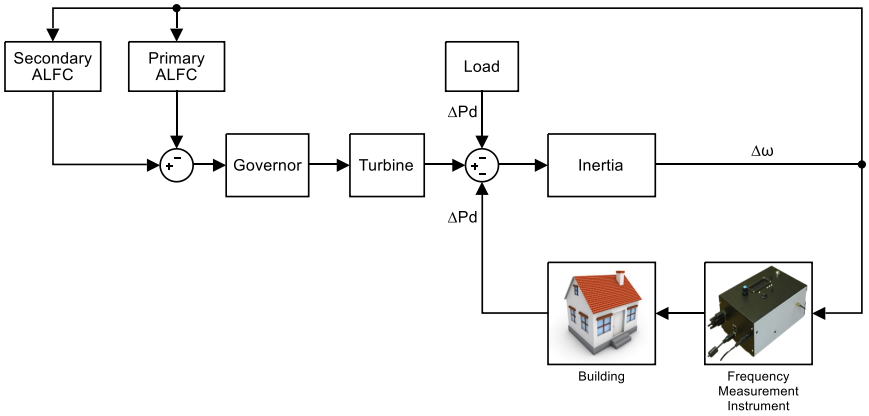


Figure 3.1 A proposed role for DFC-Primary control

3.2.2 Grid frequency

Transmission network operators in the UK are learning the consequences of the open access paradigm. An elaborate energy landscape means the demands placed on the system in terms of capacity and diversity are very different from those initially envisaged during its inception. Excessive stress and sudden natural or malicious physical events on modern power systems may degrade grid reliability and stability. Frequency stability refers to the ability of a power system to maintain frequency equilibrium following an imbalance between supply and demand. Instability occurs in the form of sustained frequency excursions which lead to tripping of generating units or loads. Therefore, grid frequency measurement provides system operators with a good indicator of system status and performance. Suppose the supply is higher than demand, grid frequency measurement increases and vice versa. In the UK, the ESO is responsible for maintaining a target frequency of the total system, which is nominally 50 Hz and controlled within the limits of 49.8 Hz to 50.2 Hz [176]. The statutory requirement permits a variation not exceeding 1% above or below 50 Hz. When the frequency exceeds these limits due to changes in supply and demand, events occur, which means power output is altered to correct the imbalances.

Figure 3.2 introduces a typical grid frequency data plot for Great Britain. The example shown starts 1st January 2018 00:00:00 and plots the measured grid frequency at a re-sampled rate of 10 min for 4.166 days. Source data is readily available in the public domain at a sample rate of 1 sec [177].

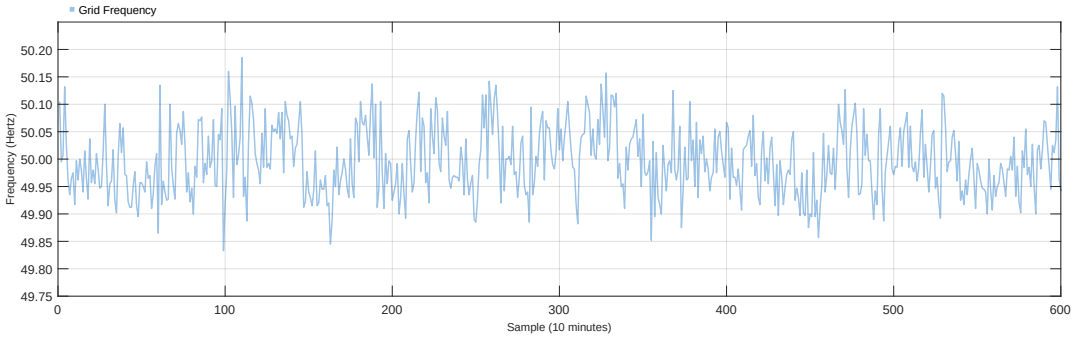


Figure 3.2 Grid frequency data for Great Britain

A useful insight into the number of events from 1st January 2015 to 31st December 2019 is shown in Figure 3.3. The number of frequency excursions less than 49.8 Hz (Lf) and the number of excursions that exceed 50.2 Hz (Hf) are shown (frequency sample rate is 1 sec). The associated first-order polynomial trendline (Lf trend and Hf trend respectively) indicate a progressive increase in the number of events over the 5 yr period. It is observed that there are no instances where the frequency exceeds the statutory requirement (50 Hz plus or minus 0.5 Hz). The number of registered Hf is notably high in October 2017 and March 2018, and Lf in July 2019. A similar analysis of frequency events in Great Britain from 2014 to 2018 concluded that the rise in the number of measured events is attributed to an increase in DREG on the UK grid [178].

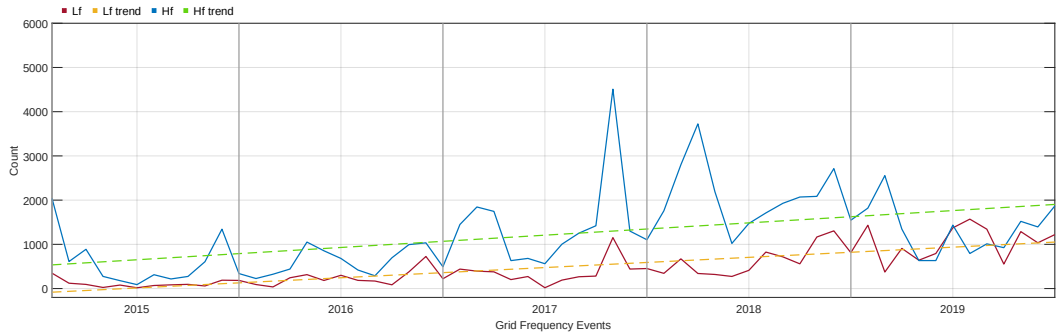
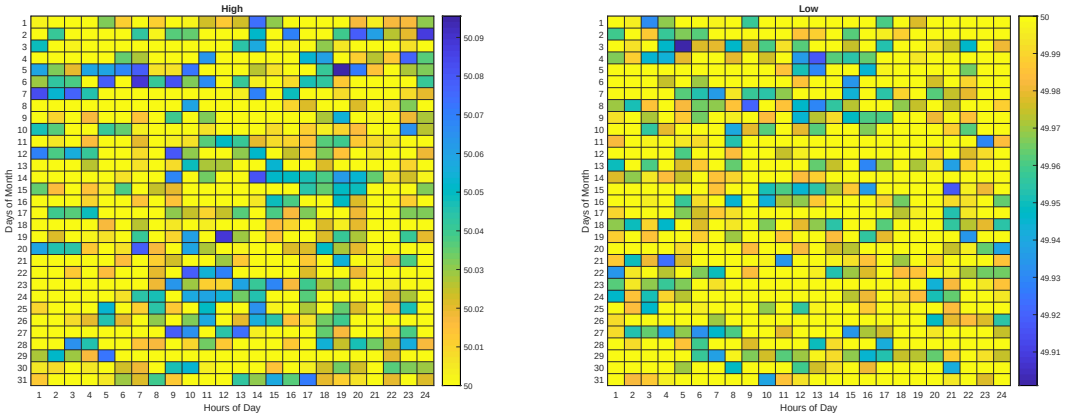


Figure 3.3 Grid frequency events 2015 to 2019

A more granular view of frequency activity is shown in Figure 3.4. Here, frequency measured at the same time of each day in January 2018 highlights the full extent of frequency deviation. The dark blue colour squares indicating a more significant departure from the nominal 50 Hz.



(a) High: $f \geq 50$ Hz **(b)** Low: $f \leq 50$ Hz

Figure 3.4 Grid frequency distribution in January 2018

To conclude this insight into grid frequency, Figure 3.5 shows grid frequency and demand data over four days starting 1st January 2018. In the later stages of this research, much work was carried out investigating the relationship between grid frequency and demand. Here, we introduce one enquiry that compared a normalised grid frequency infinite impulse response (IIR) filter output with demand data. Zero crossing points are marked, and calculated duration (D), mean (M) and area (A) values between successive points are tagged to each peak above and below 50 Hz (normalised).

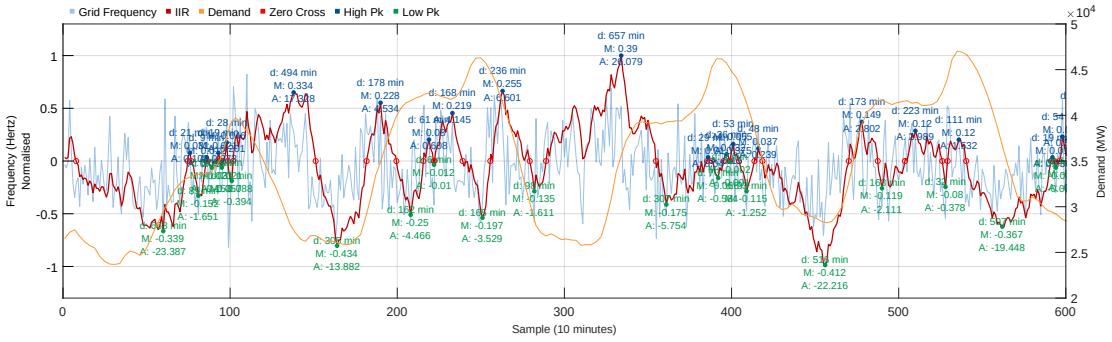


Figure 3.5 Grid frequency zero crossing

Finally, Figure 3.6 shows another enquiry that utilises techniques more commonly used in time series data mining. The alternative forms of data representation shown have been investigated because of the unique characteristics of the grid frequency, such as the large volume of data (high dimensionality) and non-linear relationships of the data elements. The time series is represented by piecewise aggregated approximation (PAA) and symbolic

aggregated approximation (SAX). The objective is to seek patterns in the data structure. A coded message is used to represent the length of the time series into a collection of strings. For example, 8LE, where 8 indicates the value is in bin 8, L indicates an increase of +3 from the previous reading and E indicates the present value duration is equivalent to two-time units (40 min). String manipulation algorithms are useful when finding patterns which can then be used for formulating prediction algorithms [179]. These techniques will be used to advance the work in subsequent chapters further.

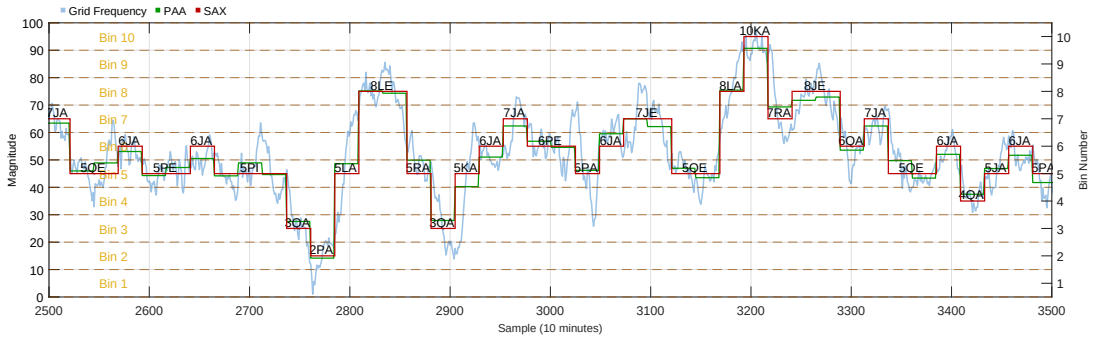


Figure 3.6 Grid frequency patterns

3.2.3 Technical development

On a synchronous generator, the frequency of the interconnected power system is implicitly instrumented via the generator tachometer, which measures shaft speed locally. At a load site, however, a dedicated means is required for frequency measurement, unless real-time communication is employed. However, the reliance on advanced communication networks to convey this information at sufficient resolution and maintain real-time accuracy at decentralised nodes is problematic [180, 181]. Many examples of frequency measurement estimation algorithms and techniques are available in the literature (e.g., [182]). However, in practice, the implementation of decentralised frequency control is not always favourable and quite often cost-inhibitive. This research offers a low-cost working prototype using the zero-crossing detection technique, constructed using an Arduino Mega 2560. The ATmega2560 low power CMOS 8-bit microcontroller based prototype (see, [183]) is designed to measure and visualise real-time grid frequency at a resolution of 100 mHz at least once every second. Figure 3.7 illustrates a breakout view of the signal conditioning and frequency measurement modules.

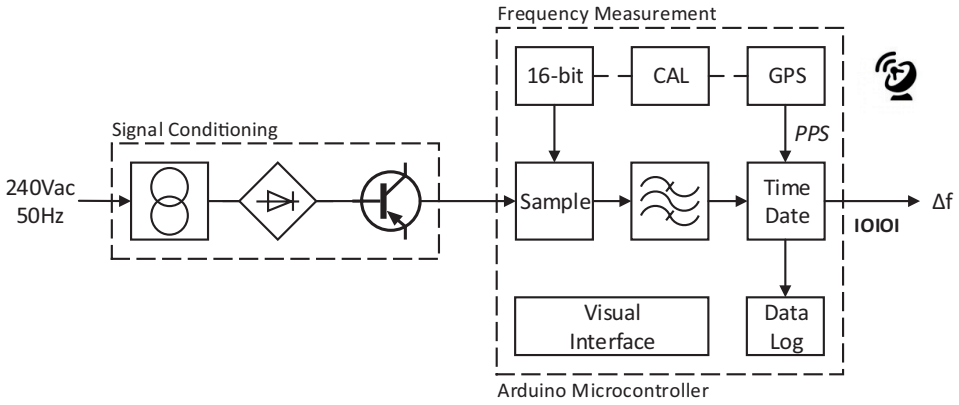


Figure 3.7 Frequency measurement instrument breakout

The prototype construction is housed inside a robust protective case for durability. An image of the prototype is shown in Figure 3.8. The design includes additional features considered appropriate to assess the performance of the instrument during subsequent analysis. For example, visual warnings are provided to highlight when specific parameters exceed set threshold values. As a consequence, the physical size of the protective case is purposely oversized to accommodate features appropriate to the prototype. Details of design construction including a catalogue of parts, visual display, and controls layout, wiring schematics and breakout pinout are documented at Appendix A.

Figure 3.8 Frequency measurement instrument

The design includes a GPS shield providing a time-base that is synchronised to coordinated universal time (UTC). In addition to geographical positional information, a GPS 1-pulse per second digital signal allows highly accurate time-date stamping for data collection, including self-calibration of the device when a GPS fix is established. The latter is necessary to

maintain measurement accuracy despite imperfections and temperature sensitivity of the microcontroller crystal oscillator. A local communications option in the form of a TIA-232 serial data interface allows the device output to be shared with a BEM system. An internal 8 GB micro secure digital (SD) card or external memory device can be used to record and store information for subsequent analysis. Real-time readout of frequency measurement and visual warnings are provided using a 16 × 2 matrix LCD and series of LED.

3.2.4 Software code development

An incremental and iterative software development process is followed during software development, providing a flexible but sufficiently robust framework to allow new features to be introduced at each stage of development. The Arduino platform connects to a bespoke integrated development environment (IDE) to upload programs (sketches) and provides access to a serial monitor console that displays text output by the Arduino software IDE. Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this section, we present the services that describe the interaction of the frequency measurement instrument and external conditions. The services provide a convenient framework to evaluate the performance of the instrument. A prototype frequency measurement instrument constructed using low-cost equipment has been introduced. Construction details are documented in Appendix A. Designed to measure and visualise real-time grid frequency at a resolution of 100 mHz at least once every second. The prototype offers and number of services. Software code has been written, which includes a series of functions designed to deliver each of the services listed in Table 3.1.

Table 3.1
Arduino software services

Service	Description
1	Measure UK grid electrical frequency at 10 mHz or 100 mHz resolution
2	Write to data log using fixed format at user-defined intervals
3	Write to Desktop PC Win10 Pro using PCI-DAS6014 interface
4	Write to LEM (Ruggedcom RX1400) using TIA-232 serial data interface
5	Display GPS location (lat/lon/alt) when GPS fix is accepted
6	Monitor and record grid frequency events
7	Power-saving features including LCD backlight auto time out
8	Turn ON/OFF write to micro SD Card
9	View data log file size (only when writing to SD Card is selected to OFF)
10	Output display refreshed at 1 sec intervals

A schematic diagram of the Arduino sketch `freq_meas_tool.ino` development with external inputs and outputs is shown in Figure 3.9.

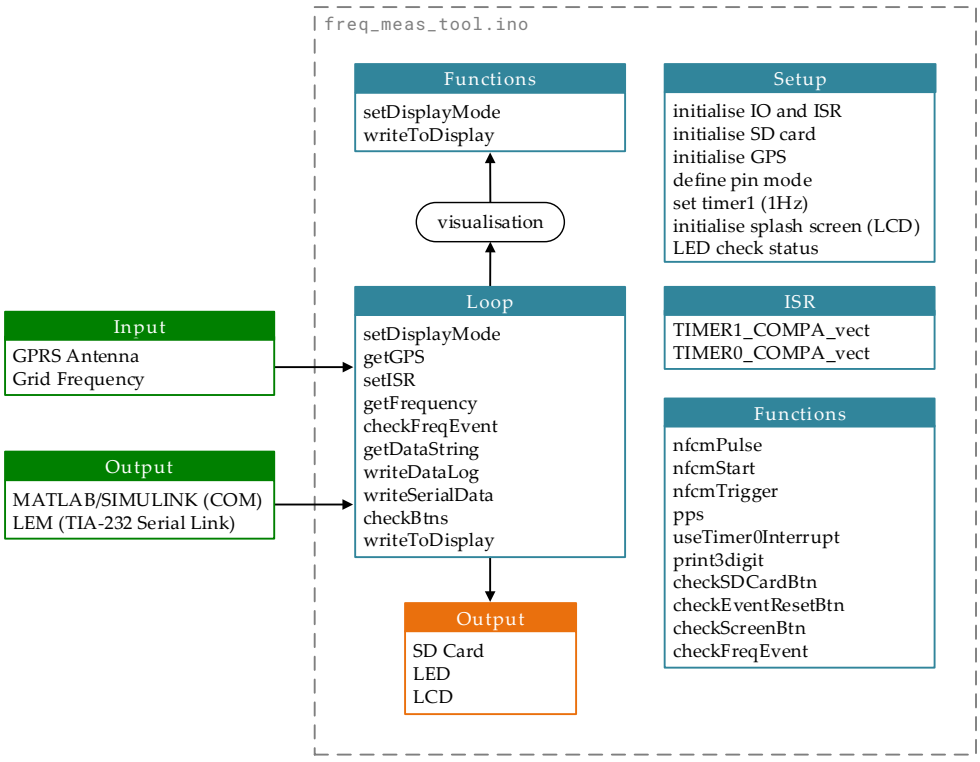


Figure 3.9 Schematic diagram of Arduino sketch development - freq_meas_tool.ino

3.2.5 System output specification

The frequency measurement instrument will deposit data summaries to two devices: (1) an 8 GB internal standard SD card for off-line data evaluation and analysis, and (2) a Ruggedcom RX1400 device, which hosts a Local Energy Management (LEM) software solution[1]. The LEM should monitor the local grid frequency at a sample rate of 1 sec. The data is time-stamped and formatted to ISO-8601 standard [184], representing times based on UTC timezone, for subsequent ingest and processing by the LEM. For consistency, the data stored to the internal SD card follows the same protocols. The continuous data string output specification is defined as `<yyyy-mm-ddThh:mm:ssZff.ff>`, where `T` and `Z` are data partition markers and `ff.ff` represents the grid frequency measurement at recorded date-time group `yyyy-mm-dd hh:mm:ss`. Figure 3.10 shows an extract of the continuous data string output from the frequency measurement instrument to the LEM.

```
...<2016-12-14T08:52:12Z49.98><2016-12-14T08:52:13Z49.98><2016-12-14T08:52:14Z49.97>
   <2016-12-14T08:52:15Z49.97><2016-12-14T08:52:16Z49.97><2016-12-14T08:52:17Z49.97>
   <2016-12-14T08:52:18Z49.97><2016-12-14T08:53:40Z50.00><2016-12-14T08:53:41Z49.95>
   <2016-12-14T08:53:42Z49.94>...
```

Figure 3.10 Data string output example

3.2.6 Baseline and performance indices

After construction, inferential statistics is used to test the hypothesis that the Arduino frequency measurement data is as good as data accessible from the National Grid. To determine whether the population variances are equal, a two-sample F-test is used for comparing two population variances σ_1^2 and σ_2^2 when a large sample (at least 30) is randomly selected from each population and the samples are independent. The **test statistic** is,

$$F = \frac{s_1^2}{s_2^2} \tag{3.1}$$

[1]The prototype frequency measurement instrument provided grid frequency measurement as part of EU H2020 funded innovation project: Demand Response in Blocks of Building (DR BOB) Teesside University, UK demonstration site (Grant Agreement No 696114).

where s_1^2 and s_2^2 represent the sample variances with $s_1^2 \geq s_2^2$. The numerator has $\mathrm{df_N} = n_1 - 1$ degrees of freedom and the denominator has $\mathrm{df_D} = n_2 - 1$ degrees of freedom, where n_1 is the size of the sample having variance s_1^2 and n_2 is the size of the sample having variance s_2^2.

Also, a two-sample z-test is performed for assessing the difference between two population means μ_1 and μ_2 when a large sample (at least 30) is randomly selected from each population and the samples are independent. The **test statistic** is $\overline{x_1} - \overline{x_2}$, and the standardised test statistic takes the form,

$$z = \frac{(\text{observed difference}) - (\text{hypothesized difference})}{\text{standard error}}$$

That is,

$$z = \frac{(\overline{x}_1 - \overline{x}_2) - (\mu_1 - \mu_2)}{\sigma_{x_1 - x_2}} \tag{3.2}$$

where

$$\sigma_{x_1 - x_2} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} \tag{3.3}$$

Given a sample size $n = 239$, s_1^2 and s_2^2 have been used in place of σ_1^2 and σ_2^2.

3.3 Simulation model development

A simulation model to determine the grid frequency response of a simplified linear power system model is created using Simulink® [185]; sample rate $Ts = 0.02$. For illustrative purposes, a single area system driven by a lumped parameter non-reheat steam turbine was implemented [186, 187]. The primary loop of the ALFC system is closed by modeling the behaviour of the power system assuming the system is operating in its normal state with complete power balance. The chief objective of the primary ALFC loop using the speed governing system is to execute the desired regulatory control on the MW output of the generators. Here, the regulation R may be expressed in per unit as well as in Hz/MW and is simply the magnitude of the slope of the speed vs. power output characteristic of the alternator. Therefore,

$$R = \frac{f_{final} - f_{initial}}{\Delta P_g} \text{ Hz/p.u. MW} \tag{3.4}$$

where P_g being the generator output power. It is noted a reduction of R results in lower static frequency drop as well as faster transient response. In addition, a supplementary loop ensures the restoration of the frequency to the nominal value. This objective is met by using an integral controller which makes the frequency deviation zero. In this context a ALFC secondary loop gain is manually adjusted through experimentation to $Ki = 0.2$ for satisfactory response in terms of overshoot and settling time.

The simulation model assumes the system is operating initially in its normal state with complete power balance and that the change in frequency is uniform. In the event of a load disturbance, it can be inferred that the system operating frequency will be less that the nominal value at equilibrium. However, from a stability perspective, the secondary loop is employed to decrease the frequency drift down to zero or to a level acceptable for stable operation.

The primary control instrument design assumes the control signal from the regulator is employed to actuate a TCL which either heats or cools the area under consideration. Also, in the context of this analysis, the control mechanism excludes any additional heat source that might compensate for any deviations of measured temperature that might otherwise compromise the occupancy comfort. Detailed models of TCL and building thermal behaviour can be found in the literature (e.g., [166, 188]). They are usually based on physical principles of mass, energy and momentum transfer and consist of complex partial differential equations that capture the building thermal and physical characteristics. However, in practice, simplified first-order models can perform just as well as more complicated models [189]. Here, the dynamic relationship between the electrical power delivered to the electro-thermal converter $P_{hp}(t)$ and the temperature of the heated zone $T(t)$, can be very well approximated by a first-order plus dead time (FOPDT) transfer function. The transfer function describes deviations away from a nominal input/output steady-state operating point. Using per-unit representations for $P_{hp}(t)$ and $T(t)$ to eliminate the steady-state gain in the model, and using Δ to represent deviations, the transfer function for the thermal response becomes:

$$G(s) = \frac{\Delta T(s)}{\Delta P_{hp}(s)} = \frac{e^{-sd}}{1+\tau s} \tag{3.5}$$

where d represents the model delay time and τ the time constant. The time constant can be assumed to be 10 to 30 min and the delay time between 0 and 5 min for a typical building [190]. A PT326 Process Trainer imitates everyday industrial situations in which temperature control is required [191]. When using this hardware to emulate building thermal behaviour, delay time (d) and process time constant (τ) measurements recorded during an open-loop step test were used to implement a proportional-integral (PI) controller following a Lambda tuning methodology [192]. This form of internal model control (IMC) (see, e.g., [193]) completes a setpoint change in about 4λ sec when operating in closed-loop mode, without overshoot, where $\lambda = (t_s - d)/4.6$, settling time $t_s = 50$, and d is the TCL (PT326 Process Trainer) calculated transport delay.

The contribution of the proposed DFC-Primary regulator is possible using this simplified thermodynamic model of a building thermal control system. In this context, assuming that the action of the speed governor plus the turbine generator is instantaneous compared with the rest of the system, it is established through experimentation, setting the regulator to the same value as the speed regulation R works satisfactory in response to a change in load. To characterise the extent of a simulated secondary DSR in a system complete with a DFC-Primary regulator, an idealised secondary demand response command event (network latency is assumed to be zero) was introduced during further tests.

With a computer model of a simplified linear power system and building thermal control system complete, a contingency load is introduced when the balance in supply and demand is at equilibrium, and the frequency is at a nominal 50 Hz and steady-state frequency error zero. In all cases, the simulations were carried out using time constants and other parameters taken from representative sources, detailed in Table 3.2, under the assumption that a significant step-change in power distribution (ΔPd) occurred at the beginning of each simulation. For example, the impact on steady-state frequency deviation (Δf_{ss}) of -0.01212 p.u. is calculated at Equation (3.6) when a demand side load (ΔPd) of 75 MW is applied; representing the loss of a medium-sized generator on the supply side. The model (including DFC-Primary regulator) is shown in Figure 3.11.

$$\Delta f_{ss} = \frac{-\Delta Pd}{D + \frac{1}{R}} = \frac{-0.25}{0.8 + \frac{1}{0.05}} = -0.01202 \text{ p.u.} \tag{3.6}$$

Table 3.2
Decentralised primary frequency control model parameters

Name	Parameter
Power system rating	300 MVA
Nominal frequency (f)	50 Hz
Nominal demand load (Pd)	75 MW
ALFC secondary loop gain (Ki)	0.2 p.u. MW/Hz s
Speed regulator (R)	0.05 Hz/p.u. MW
Inertia time constant (H)	5 sec
Load damping constant (D)	0.8 sec
Governor time constant (Tg)	0.25 sec
Turbine time constant (Tt)	0.60 sec
DFC-Primary regulator (R)	0.05
Thermal load time constant (Th)	9.65 sec
FOPDT thermal load gain (Kh)	1.16
FOPDT transport delay (d)	0.45
TCL controller proportional gain (P)	0.51750
TCL controller integral gain (I)	0.10363
Temperature setpoint ($OF1$)	7.5 ($\simeq 40\,^{\circ}$C)
Secondary Demand Response gain (SDR)	0.5
Calibration factor (electrical power) ($OF2$)	0.274
Calibration factor (temperature) ($OF3$)	0.345
Compensator gain ($G1$)	20
Compensator gain ($G2, G3$)	$1/G1$

In the model illustrated in Figure 3.11, a power system rating of 300 MVA is assumed. A contingency of 75 MW power disturbance ($\Delta Pd = 75$ MW, 0.25 p.u.) step response at $t = 750$ sec was introduced, allowing time for the system to initialise, before triggering a change in frequency.

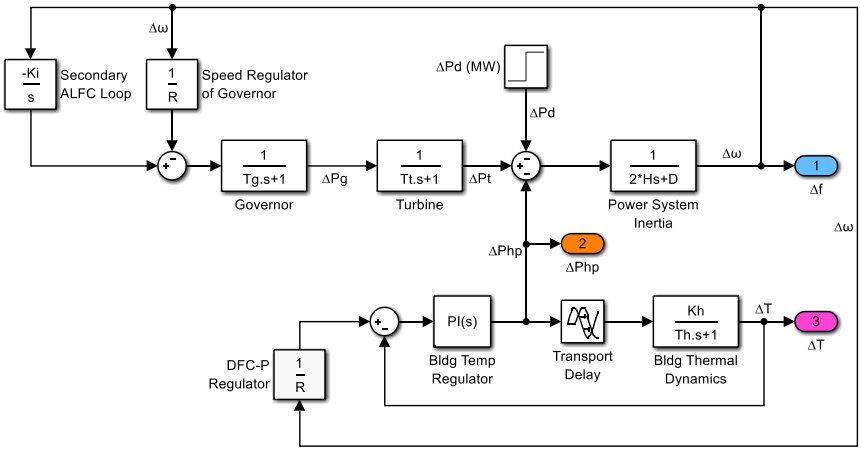


Figure 3.11 Simulink® model of decentralised primary frequency control

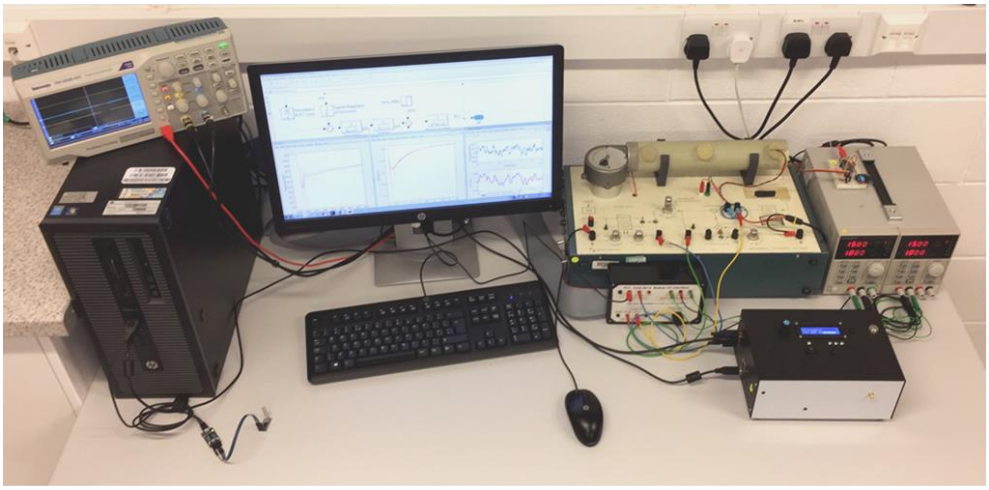
3.4 Experimental test design and setup

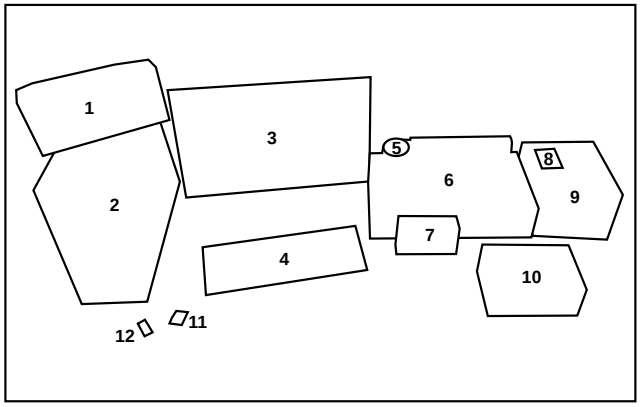
Low-level software testing designed to validate specific methods and functions is carried out during each stage of writing software code. Subsequent integration tests that verify the interaction with other services is then carried out before attention focuses on performing functional tests. Here, we test for specific criteria. The final tests aim to replicate user behaviour with the frequency measurement instrument in its application environment. In this work, we present two experimental tests:

1. **Thermostatically controlled load test**. This test removes the simulated thermal load with hardware designed to heat air that is drawn from the atmosphere by a centrifugal blower. Afterwards, the warm air is released back into the atmosphere through a duct which houses a temperature sensor.

2. **End-to-end test**. End-to-end testing is a methodology that validates an application workflow from start to finish by simulating real use scenarios. This test expands on the previous testing and aims to verify data flow and temperature regulation by combining the process control loops encoded in the simulation model, PT326 Process Trainer and the frequency measurement instrument. These individual assets are grouped such that we can monitor a change in temperature when the simple closed-loop control system reference input is the measured grid frequency data stream.

The first test is designed to replace the FOPDT transfer function with a PT326 Process Trainer to simulate the dynamic relationship between the electrical power delivered to the electro-thermal converter and the temperature of the building. In the second test, a continuous input stream of real-time grid frequency measurement output from the Arduino frequency measurement instrument is introduced. An image showing the hardware equipment configuration setup for both tests is shown in Figure 3.12.





Item	Description
1	Oscilloscope
2	Desktop PC Win10Pro
3	Monitor
4	Keyboard
5	Heater
6	PT326 Process Trainer (TCL)
7	PCI-DAS6014 Interface
8	DHT22 Temperature Sensor
9	Power Supply Unit
10	Arduino Frequency Sensor
11	433MHz RF
12	USB UART

Figure 3.12 Decentralised frequency control test environment

3.4.1 Thermostatically controlled load

A PT326 Process Training replaces the simulated building thermal dynamics represented by the transfer function:

$$G(s) = \frac{Khe^{-sd}}{Th(s)+1} \tag{3.7}$$

where $Kh = 1.16$, $Th = 9.65$ and $d = 0.45$. The PT326 heats air drawn from the immediate space by a centrifugal blower and is heated as it passes over a heater before it is then released back to the local space through a duct. A PCI-DAS6014 hardware item is configured to provide an interface between the on-bench TCL and PI controller (Figure 3.13). The PT326 mass flow air temperature is measured using a thermistor placed at a position such that the spatial separation between the heater coil and thermistor introduced a transport delay $d = 0.45$ into the system. In practice this time delay is much greater, typically 5 to 10 min depending on a multitude of factors. A secondary advisory temperature measurement taken from an Arduino compatible temperature and humidity sensor (DHT22) is positioned directly into the PT326 mass airflow outlet. A 433 MHz RF communication network is established between the remote DHT22 sensor and Simulink® model to enable the observer to record temperature data during each test. Here we use an Arduino Nano to interface between the temperature sensor and a 433 MHz transmitter. To enable the desktop PC to communicate with the microcontroller a 433 MHz receiver is connected to a universal serial bus (USB) to universal asynchronous receiver/transmitter (UART) converter.

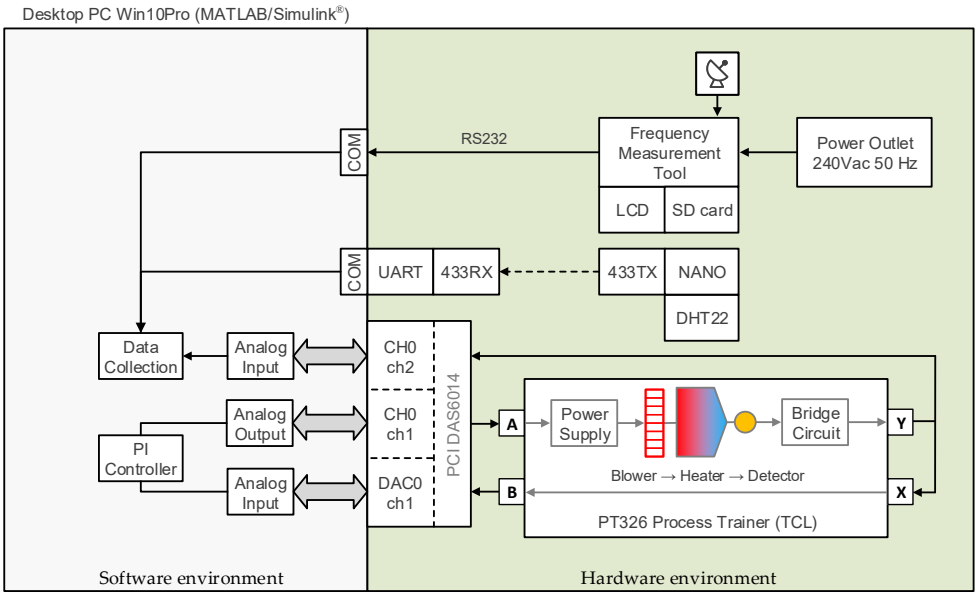


Figure 3.13 Simulation model and PT326 hardware interface

A simulation test to determine the grid frequency and external TCL (PT326 Process Trainer) temperature response is configured using Simulink® software. A modified simulation model includes additional library blocks that are designed to provide an interface between the

model and PCI-DAS6014 (Figure 3.14). After calibration, the PT326 temperature setpoint (OF1) is set and maintained at 7.5. The temperature gauge registered a value of 40 °C, i.e., mid-scale. A contingency of 75 MW power disturbance ($\Delta Pd = 75$ MW, 0.25 p.u.) step response at $t = 750$ sec is introduced, allowing time for the system to initialise, before triggering a change in frequency. Results will show that a simulated frequency response $\Delta f(t)$, recorded variation in temperature $\Delta T(t)$ and electrical power $\Delta P_{hp}(t)$ delivered to the electro-thermal converter (Figure 3.11). When combined with experimental tests, these results demonstrates that the performance of decentralised frequency control is credible.

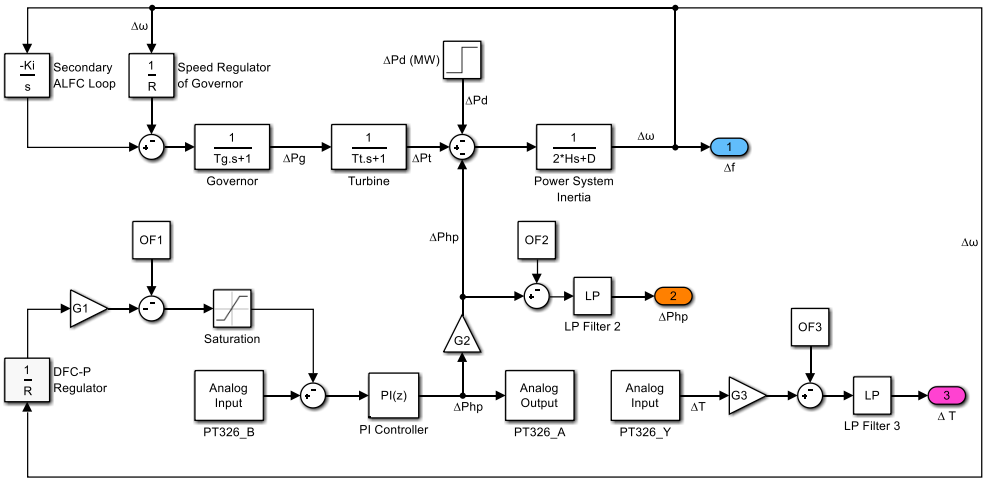


Figure 3.14 A modified simulation model for loop frequency test

3.4.2 End-to-end test

The second experimental test requires further modification to the thermostatically controlled load test (Section 3.4.1). The simulation model (Figure 3.13) is revised. Simulation blocks that imitate a single area power system driven by a lumped parameter non-reheat turbine steam turbine are replaced with new blocks that provide an interface between the frequency measurement instrument and the simulation model. Both the frequency measurement instrument and PT326 Processes Trainer are now physically connected to the desktop PC. Connecting the frequency measurement instrument to a standard UK single-phase 3-pin AC power outlet (240 VAC 50 Hz), measured grid frequency is streamed into the model using the RS232 connection between frequency measurement instrument and desktop PC. The FOPDT transfer function representing the building thermal characteristics, substituted for

the PT326 Process Trainer in earlier tests, remains unchanged. Thus, the Simulink® model can provide closed-loop control of the heater, which is regulated by the input stream of the measured grid frequency. Figure 3.15 shows the experimental test configuration set up.

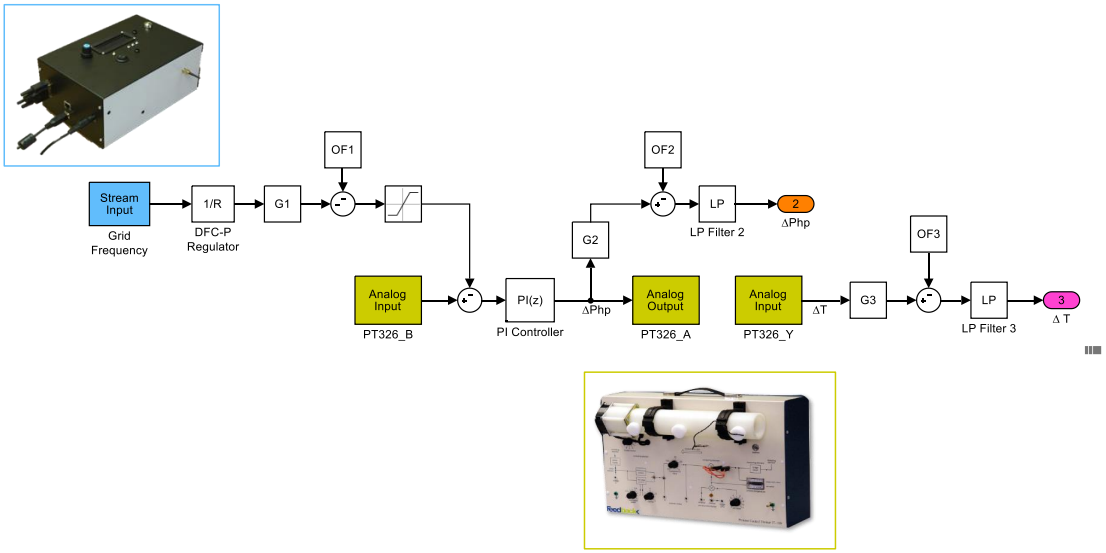
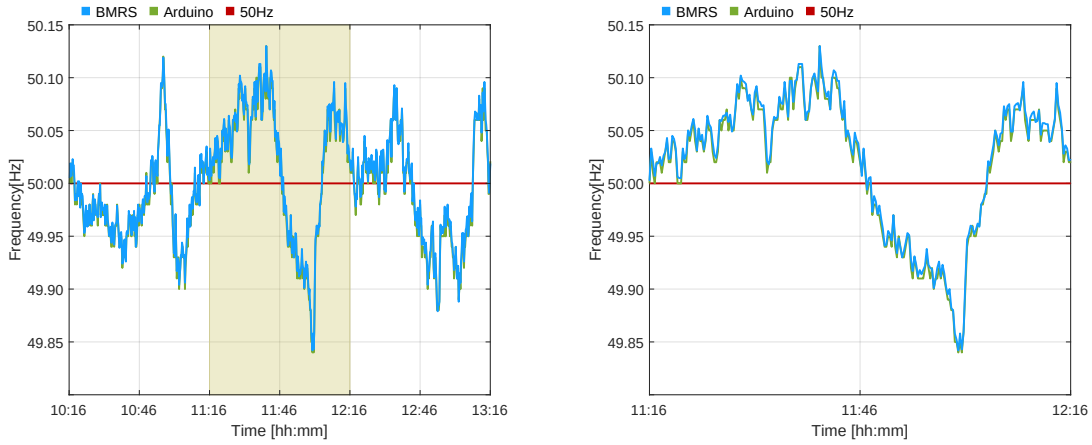


Figure 3.15 A modified simulation model for end-to-end test

3.5 Frequency control regulation

3.5.1 Microcontroller frequency measurement instrument

A graphical view of frequency data recorded using the microcontroller appears to exhibit similarities when compared to BMRS data Figure 3.16. However, it is inferential statistics that provide the necessary toolset to help us draw conclusions about a population (the collection of outcomes) to verify the device output performance based on probability [194]. In the following sections, tests for homogeneity of variances are adopted when comparing the normal population using equal size data from two independent samples (Arduino sensor data sample and BMRS data sample).



(a) 3 hr **(b)** 1 hr magnified

Figure 3.16 Frequency measurement instrument and BMRS data at 15-Dec-2017 10:16 at 15 sec resolution

Performing two-sample f-test for variance and two-sample z-test for difference between means was conducted using random sample data recorded at a 15-sec resolution for a 60 min period commencing 14:23:00 on 14th December 2016. Sample 1 (s_1) is derived from the Arduino microcontroller frequency sensor, and Sample 2 (s_2) is historical data of the same resolution and time-period obtained from BMRS. Both samples consist of 239 recordings. The objective is to provide quantitative evidence to substantiate a claim that the frequency measurement recorded using the Arduino based frequency sensor is as good as grid frequency data accessible from the National Grid.

3.5.2 Two-sample f-test for variance

From Table 3.3 $s_1^2 = 0.00059703$ and $s_2^2 = 0.00058566$, therefore given $s_1^2 > s_2^2$ we declare s_1^2 and σ_1^2 can be used to represent the sample and population variances for the Arduino frequency measurement instrument, respectively.

Table 3.3
Sample variance and standard deviation

Sample	Variance (s^2)	Std Deviation (s)
1	0.00059703	0.0242
2	0.00058566	0.0244

With the claim '*The Arduino microcontroller frequency sensor measured mains frequency is as good as the BMRS grid frequency data*' the null and alternative hypothesis are $H_o : \sigma_1^2 = \sigma_2^2$ and $H_a : \sigma_1^2 \neq \sigma_2^2$ (claim). Noting the test is two-tailed, given a significance value where $\alpha = 0.05$, then $1/2\alpha = 1/2(0.05) = 0.025$, the degrees of freedom $\mathrm{df_N} = n_1 - 1 = 239 - 1 = 238$, and $\mathrm{df_D} = n_2 - 1 = 239 - 1 = 238$, the critical value is $F_o = 1.2901$. So, the rejection region is $F > 1.2901$, i.e., you would observe values greater than 1.2901, only 5% of the time by chance. Before deciding to reject or fail, the null hypothesis the test statistic, F is calculated, such that $F = (s_1^2)/(s_2^2) = 0.00059703/0.00058566 \approx 1.0194$. Because F is not in the rejection region, the decision is to fail to reject the null hypothesis. For this reason, it is possible to claim there is no significant evidence to reject the null hypothesis and therefore conclude there is no significant difference between the methods presented when measuring mains frequency.

3.5.3 Two-sample z-test for difference between means

A similar approach is now followed to test the difference between means. The claim '*There is no difference in the mean grid frequency recordings of the Arduino microcontroller frequency measurement instrument and data from BMRS*' so the null and alternative hypothesis are $H_o : \mu_1^2 = \mu_2^2$ and $H_a : \mu_1^2 \neq \mu_2^2$. The difference of significance, α is given $1/2\alpha = 1/2(0.05) = 0.025$, $\mathrm{df_N} = 238$ and $\mathrm{df_D} = 238$, and the critical value $-z_0 = -1.96$ and $z_0 = 1.96$ established, in this instance, by executing MS Excel Data Analysis Tool > z-Test: Two Sample for Means. Given the rejection regions are $z < -1.96$ and $z > 1.96$ and both samples are large (> 30), s_1^2 and s_2^2 can be used in place of σ_1 and σ_2 to calculate the standard error $\sigma_{\overline{x}_1 - \overline{x}_2} \approx 0.002225$; Equation (3.3). This result is used to determine the standardised test statistic such that $z = -1.824472$; Equation (3.2).

The graph shown in Figure 3.17 shows the location of the rejection regions and the standardised test statistic z. Because z is not in the rejection region, there is not enough evidence at the 5% level of significance to support the claim that there is a difference in the mean grid frequency recordings of the Arduino microcontroller frequency measurement instrument and data taken from BMRS. The decision is to fail to reject the null hypothesis.

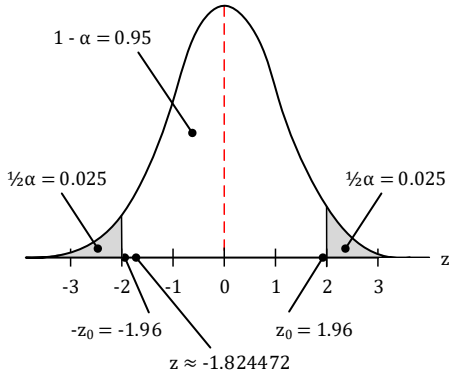


Figure 3.17 Rejection regions and standard test statistic z

The results of both hypothesis tests provide enough evidence to support a claim that there is no difference in the variance and mean frequency sensor values recorded using the Arduino microcontroller frequency measurement instrument when compared against the values obtained from BMRS (see Figure 3.18). This interpretation based on inferential statistical analysis supports the use of the low-cost frequency measurement instrument as part of the proposed decentralised primary frequency control strategy.

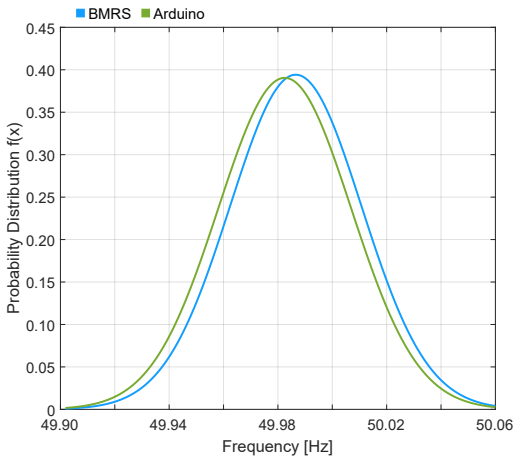


Figure 3.18 Distribution plot two-sample data with equal df

3.5.4 MATLAB/Simulink® ALFC

As discussed previously, computer-based simulations were performed to validate the behaviour of a single area non-reheat steam turbine power system ALFC primary and secondary control

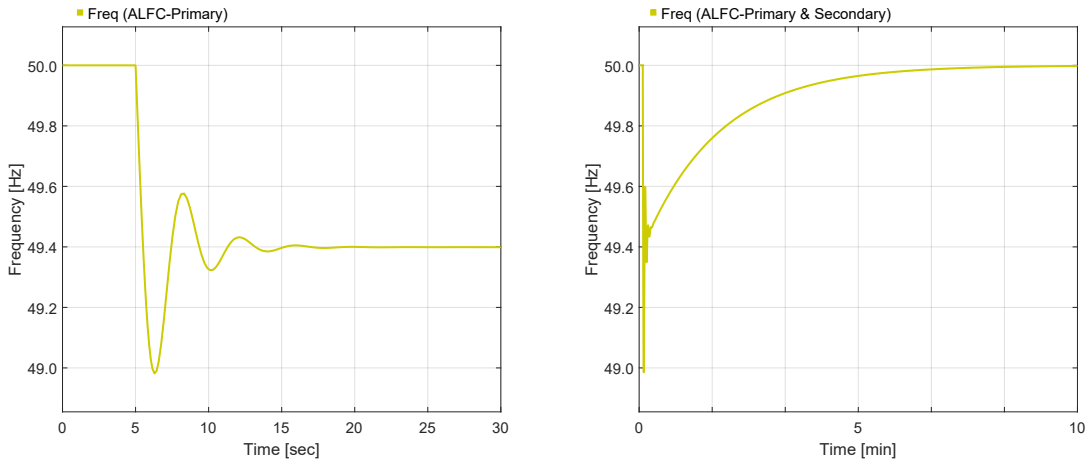
loops. In the first simulation, a demand side load (ΔPd) is applied, and the recorded frequency response checked against mathematical reasoning.

A power system rated output is 300 MVA at a nominal 50 Hz. The response of this power system to changes in load demand depends on the value of the speed governor regulator (R) and the frequency dependency of the load. Considering only the effects of the ALFC primary loop, given the power system parameters at Table 3.4, shows a calculated steady-state frequency deviation (Δf_{ss}) of -0.01202 p.u. when a demand side load (ΔPd) of 75 MW is applied. In the absence of any frequency sensitive loads, the load damping constant $D = 0$.

Table 3.4
Power system parameters

Parameter	Description	Value
H	Inertia time constant	5 sec
Tg	Governor time constant	0.25 sec
Tt	Turbine time constant	0.6 sec
R	Regulator	0.05 p.u.
D	Damping constant	0.8

The result of Equation (3.6) is equivalent to a decrease in frequency measurement $-0.01202 \times 50 = -0.601$ Hz. Figure 3.19a confirms a steady-state frequency of 49.4 Hz is attained at $t \approx 20$ sec. The effects of the ALFC secondary low-gain integrator loop when working in slow reset mode, adjusts the reference power command, thus eliminating the steady-state frequency deviation. This gradual adjustment will continue until the frequency error is zero and typically measured in minutes (Figure 3.19b).

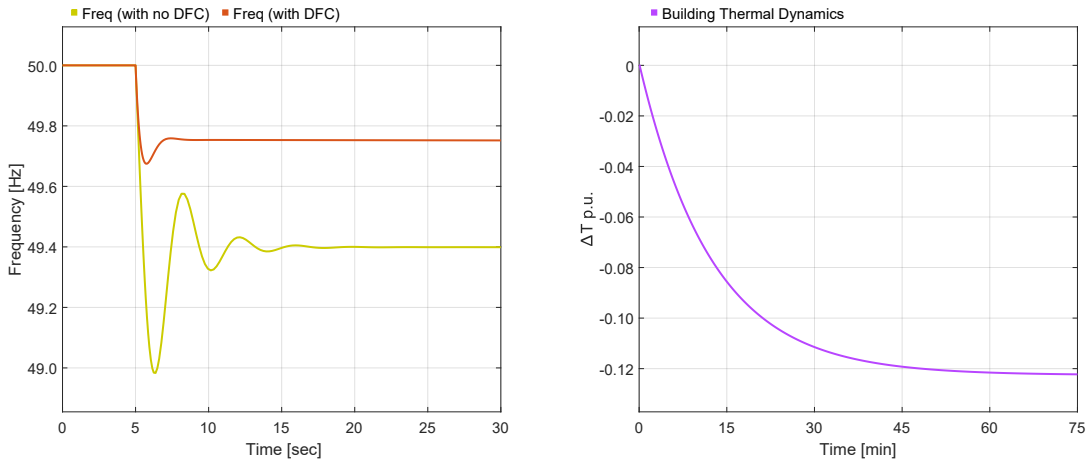


(a) ALFC primary loop **(b)** ALFC primary and secondary loop

Figure 3.19 Frequency response

3.5.5 MATLAB/Simulink® DFC-Primary

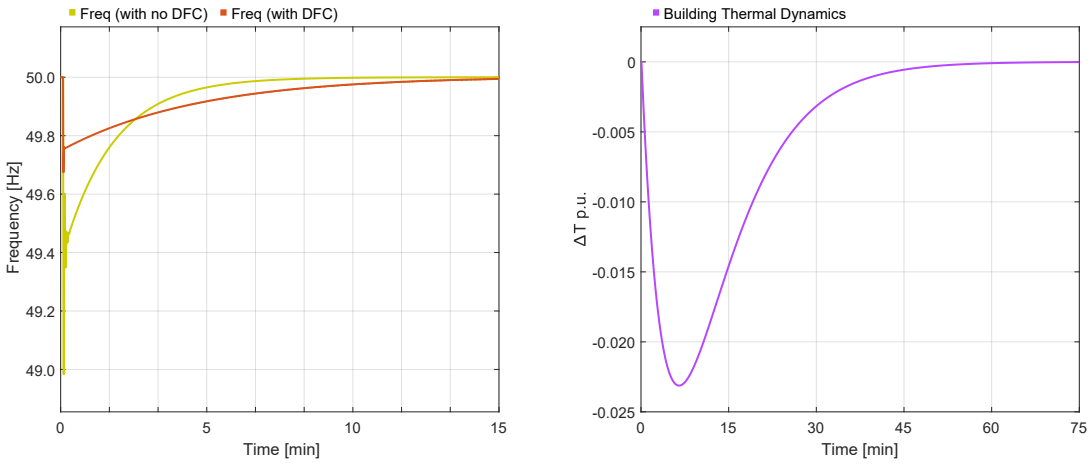
A small gain regulator is introduced to the simulation model DFC-Primary loop. The initial simulation was carried out with no ALFC secondary loop. The results shown in Figure 3.20a compares the frequency response with no DFC-Primary regulation against the frequency response of the same model with DFC-Primary control applied. The analysis shows the action of the DFC-Primary regulator when included in the model not only eliminates the observed frequency oscillation but also reduces the measured frequency deviation. Figure 3.20b shows the reduced building temperature with respect to time when a large supply side demand load ($\Delta Pd = 75$ MW) is suddenly introduced at $t = 5$ sec. As reflected the thermal inertia extends the time of change in the simulated temperature until a steady-state is reached. Clearly, some means of secondary (integral) action is required to restore equilibrium.



(a) Frequency response **(b)** Building thermal dynamics

Figure 3.20 ALFC primary loop and DFC-Primary regulator

Results of a further test cycle with ALFC secondary loop included confirms the action of the small gain integrator is consistent with earlier tests, slowly restoring the frequency deviation to zero over an extended period. When observing the graphs shown in Figure 3.21, it is evident that the transient is reduced in magnitude and is less oscillatory. However, the time taken to reach frequency equilibrium when DFC-Primary regulation is present is marginally longer when compared to the same frequency response with no DFC-Primary regulation. Also, the increased damping allows the integrator gain to be increased, lowering the restoration time without introducing oscillation. Introducing a secondary control in the form of a small gain integrator adjusting the building temperature setpoint also reduced the settling time. Figure 3.21b indicates that only a short-lived thermal transient of small magnitude and approximately twice the duration of the restoration time for electrical frequency was encountered. A priori knowledge suggests the impact of a small thermal transient such as this is likely to have little effect upon building thermal comfort. Overall, these results indicate that the proposed DFC-Primary regulator has the potential to leverage a contribution to frequency regulation of the power system. Therefore, the approach could be integrated within a traditional DR scheme, which could be implemented using low-cost embedded hardware such as an Arduino microcontroller.



(a) Frequency response **(b)** Building thermal dynamics

Figure 3.21 ALFC primary and secondary loop and DFC-Primary regulator

3.6 Summary

In this chapter, a decentralised frequency control regulation method has been validated using a series of conceptual models containing a single area power network, a standalone frequency measurement instrument and a real controllable thermal load. A FOPDT transfer function model of the thermal load was initially identified and later substituted for a PT326 Process Trainer during a series of experimental tests. The overall approach was validated by controlling the temperature evolution (and electrical load) of the trainer initially from a signal designed to imitate grid frequency, by using it as a reference signal to a closed-loop controllable load.

Several interesting conclusions can be drawn from the results presented. They suggest that small excursions in measured temperature from TCL setpoint values will not compromise indoor comfort temperatures but can contribute to the restoration of frequency equilibrium during network stress events. These findings mean that the utility of a pro-active decentralised control strategy directly could close the gap in reserve capacity margins availability by exploiting coupling technologies such as heat pumps and other TCLs with near-zero intervention from the consumer.

The construction and software development of a prototype frequency measurement instrument was configured to stream grid frequency measurements directly from a standard UK

single-phase power outlet socket (240 VAC 50 Hz) into a simulation model. This chapter has conclusively established the design and implementation means the frequency measurement instrument may have real benefit operating in standalone mode supporting demand response actions as part of a wider decentralised community power system.

Chapter 4

Electricity Demand Forecasting

4.1 Introduction

The world is experiencing a fourth industrial revolution. Rapid development of technologies is advancing smart infrastructure opportunities. Experts observe decarbonisation, digitalisation and decentralisation as the main drivers for change. In electrical power systems, a downturn of centralised conventional fossil fuel-fired power plants and increased proportion of distributed power generation adds to the already troublesome outlook for operators of low inertia energy systems. In the absence of reliable real-time demand forecasting measures, effective decentralised demand-side energy planning is often problematic. In this chapter, we formulate a simple yet highly effective lumped model for forecasting the rate at which electricity is consumed. The methodology presented focuses on the potential adoption by a regional electricity network operator with inadequate real-time energy data who requires knowledge of the wider aggregated future rate of energy consumption. Thus, contributing to a reduction in the demand for state-owned generation power plants. The forecasting session is constructed initially through analysis of a chronological sequence of discrete observations. Historical demand data shows behaviour that allows the use of dimensionality reduction techniques. Combined with piecewise interpolation, an electricity demand forecasting methodology is formulated. Solutions of short-term forecasting problems provide credible predictions for energy demand. Calculations for medium-term forecasts that extend beyond six months are also very promising. The forecasting method offers a way to advance a decentralised informatics, optimisation, and control framework for small island power systems or distributed grid-edge systems as part of an evolving demand response service.

4.2 Methodology

The proposed data-driven methodology is divided into three distinct parts (Figure 4.1). Analysis of a chronological sequence of discrete observations is first performed, and the composition of the univariate one-dimensional time series is determined. In the second step, a dimensionality reduction technique is applied before piecewise interpolation is used to smooth subsequent consecutive polynomial segments. A resultant lookup table provides the necessary metadata for the forecasting algorithm to model the demand characterisation. The objective is to maintain an accurate 4 hr electricity demand prediction horizon. However, results show this can be changed to much more extended periods while maintaining competitive results.

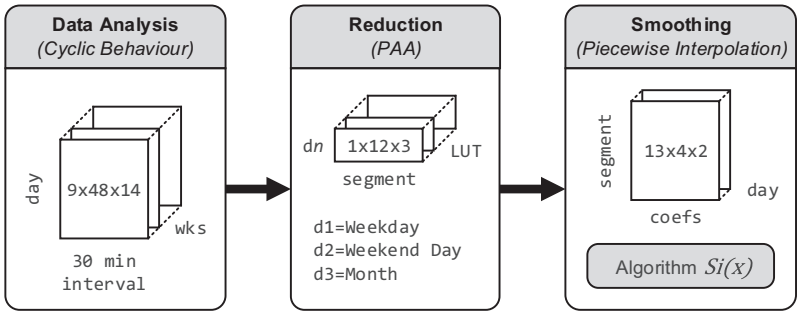
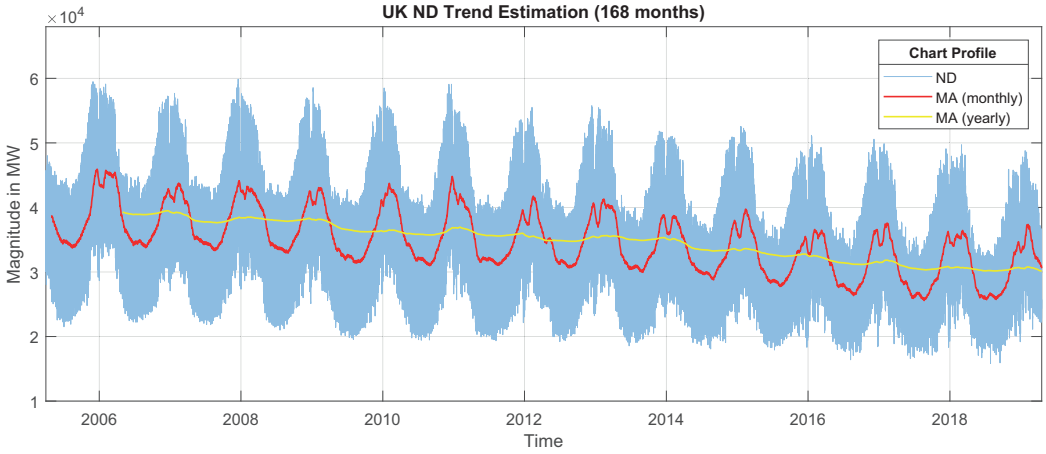


Figure 4.1 A visual representation of demand forecasting methodology

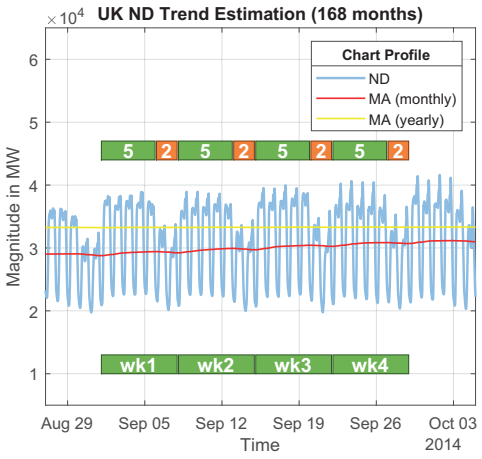
4.3 Composition of time series

The Electricity System Operator (ESO) in Great Britain publishes historic national demand data [18]. The data represents the generation requirement, which is derived from National Grid operational generation metering recorded at 30 min intervals. In this research, analysis is based on national demand data from 1st April 2005 to 31st March 2019, comprising 245,424 data items. Learning from real data is an essential attribute of most pattern recognition systems. The performance of the proposed forecasting method is validated against more recent data. The first task is to extract crucial characteristics. Time series classification is a prevalent machine learning problem widely accepted in various domains [195–197]. Often, complex time series values are converted into visual patterns which allow for the problem to be presented as image recognition problems [198]. In the big data context, pattern mining techniques such as sequential pattern mining (SPM) (see, e.g., [199]) have gained

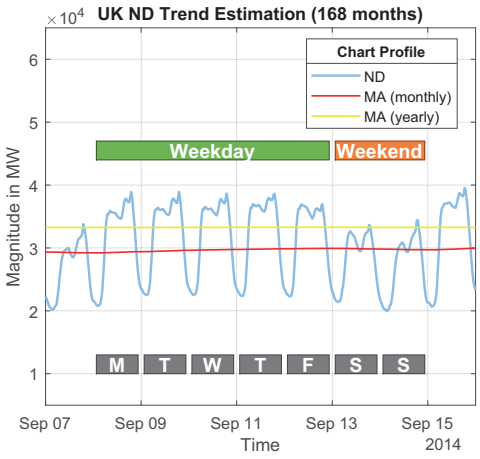
significant attention to meet the increasing demand for large-scale computing. The underlying objective is to find correlations in data. A more general-purpose strategy used in this current research relates to the use of statistical techniques for analysing data measurements to extract meaningful characteristics. Using statistical pattern recognition is used to justify design progression. The process involves three stages: (1) data acquisition and preprocessing, (2) data representation, and (3) decision making [200]. Figure 4.2a shows the complete time series data set used to create the load forecasting algorithm.



(a) Demand data 168 months



(b) Demand data 1 month



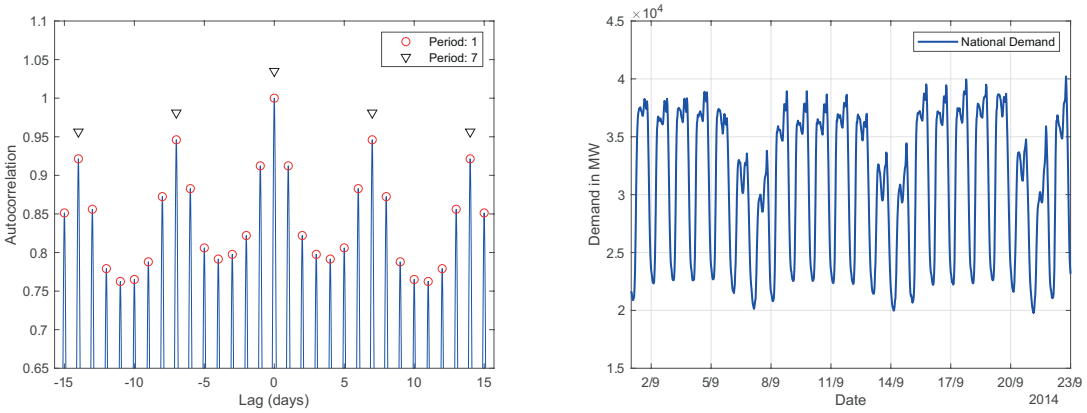
(c) Demand data 1 wk

Figure 4.2 UK National demand data
(01-Apr-2015 to 31-Mar-2019)

A simple moving average of monthly and yearly trend estimations provide a clear image of the demand data characteristics. Figure 4.2b and Figure 4.2c show distinct patterns of

regularity over a 4 wk and 1 wk periods, noting the marked difference between weekday and weekend day.

Computing the autocorrelation of the time series identifies the periodicity of the signal. Figure 4.3 shows the time between each peak is consistent with a typical weekly pattern consisting of five similar weekday oscillations followed by two weekend day oscillations, also of similar form.



(a) Autocorrelation shows weekly patterns **(b)** UK National demand profile

Figure 4.3 Composition of demand data

Regression is used to remove fluctuations in the time series and to identify potential seasonal and cyclic behaviour. The approach used to remove the trend from the time series first calculates the least squares regression line (see, [201]) before subtracting the deviations from the least squares fit line from the time series. Given the equation for a straight line is $y = bx + a$ where b is the slope of the line, and a is the y-intercept, the best fit line (regression line) for the points $(x_1, y_1), \ldots, (x_n, y_n)$ is given by $y - \overline{y} = b(x - \overline{x})$ where,

$$b = \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) / \sum_{i=1}^{n} (x_i - \overline{x})^2 \tag{4.1}$$

and the y-intercept is defined as $a = \overline{y} - b\overline{x}$. The overbar is used to denote average value. In the absence of outliners, Equation (4.2) is used to normalise the time series in which the values are shifted and rescaled so that they end up ranging between a minimum and maximum input value. It is also known as min-max scaling. In this instance we choose to

normalise the time series where the lower input range $l = 0$ and maximum of input range $u = 100$.

$$x' = l + [(x - x_{min})(u - l)]/(x_{max} - x_{min}) \tag{4.2}$$

A $9 \times 48 \times 14$ multi-dimensional array characterises 14 distinct weeks, where each week identified commences on the Monday immediately following the lowest recorded demand data in each year (2005 to 2019). Measurements recorded at 30 min intervals for each day are assigned to columns 1 to 48; the mean value of rows 1 to 5 (weekdays) and rows 6 and 7 (weekend days) are assigned to rows 8 and 9 respectively. A mean value of the collective row 8 and 9 are then computed to enumerate a generalised demand profile shape for any weekday and weekend day, respectively.

A simple moving average of order n process given at Equation (4.3) smooths the original demand data y_i; where n represents a set number of observations for one month and year, respectively.

$$y_t = \frac{1}{n} \sum_{i=t-n+1}^{t} y_i \tag{4.3}$$

Analysis reveals, in addition to daily/weekly characteristics, the time series also displays seasonality and negative secular trend with constant variability. The general idea is to define a model from historical time series that enumerates the cyclic behaviour and negative secular trend that can be used as part of the forecasting algorithm. For seasonality, the mean of each moving average 12 month period is calculated before applying a dimensionality reduction technique. Furthermore, in this strategy, the negative secular trend is expressed in mathematical terms using Equation (4.1). Here, the coefficients for a polynomial that is a best fit (least squares method) of the given set of data are calculated.

The composition of the time series observed is characterised by three seasonal patterns: weekday, weekend day and month. Given the volume of historical data available, we first present a method to reduce time series feature dimensionality and then formulate the forecast prediction algorithm.

4.4 Dimensionality reduction

Time series analysis is a statistical technique often used to analyse the pattern of discrete observations over time to forecast future events. When the number of observations is large, time series analysis becomes time-consuming. Dimensionality reduction techniques can be used to help improve the classification of big data for time series analysis, thus improving the efficiency of the forecasting process. Piecewise aggregate approximation (PAA) proposed by Keogh et al. [20] is a well-known technique that reduces the dimensionality of a time series and for data representation. We choose to approximate the data with a piecewise coefficient such that the period between each change point is 2 hr. In this method, the normalised demand time series window of size n is first divided into k segments of equal length. The average value of the data of the segments is then used as the representative value of each segment. Therefore, the demand time series PAA representation will be a k-dimensional vector $\boldsymbol{x_i} = \overline{x}_i, \ldots, \overline{x}_n$ of the mean values of each segment. The dimensionality reduction calculation is computed by Equation (4.4).

$$\overline{x}_i = \frac{k}{n} \sum_{j=\frac{n}{k}(i-1)+1}^{\frac{n}{k}i} x_j \tag{4.4}$$

Simply stated, to reduce the time series dimensionality of length n to k, the data is first divided into k equally sized segments then the mean value of the data in each segment is calculated. The subsequent vector of these values represents the reduced dimensionality of the original dataset. The effect of applying PAA to the demand data where each segment is 2 hr in duration over a 24 hr period (12 equal length segments) is shown in Figure 4.4.

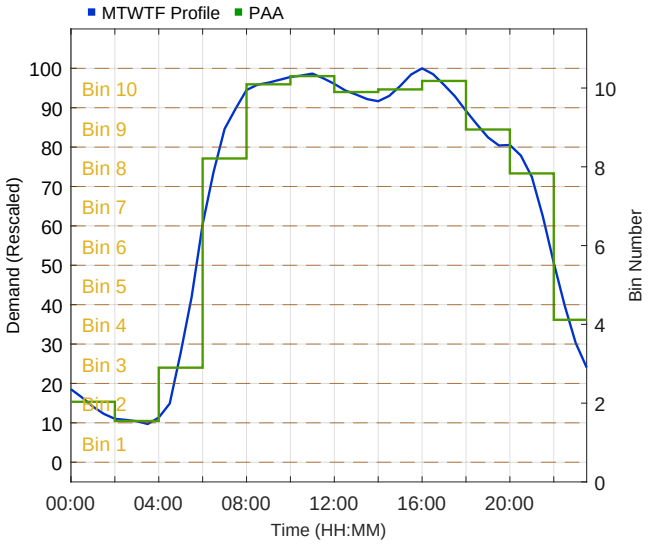


Figure 4.4 Weekday demand profile with PAA applied

The equation provides the mean of the elements in the equi-sized frames, which makes up the vector of the reduced dimensional time series. The method is applied to the *day* and *month* features. A numerical investigation comparing different piecewise coefficients confirms the dimensionality could be reduced at the same time as preserving enough information about the original data.

After the time series is transformed into segments using PAA technique, the data is discretised, grouping the continuous input into a finite number of discrete bins. The translation means the data dimensionality can be reduced further and converted into a symbol string using symbolic aggregate approximation (SAX), i.e., each region is assigned a symbol according to the determined change points. In the context of data mining, SAX is comparable to other techniques, including discrete Fourier transform and discrete wavelet transform while requiring less storage [21]. This strategy is particularly useful for low-complexity solutions, as they are less data-intensive than more complex econometric methods and models needed for forecasting [22]. In this work, the SAX symbol string (symbolic conversion) is a 4-bit binary representation of the discrete bin the continuous input was assigned after discretisation (Figure 4.5). In this instance, the length of each SAX segment is not fixed. Instead, it only changes when the value of each PAA segment exceeds an upper or lower discrete bin value.

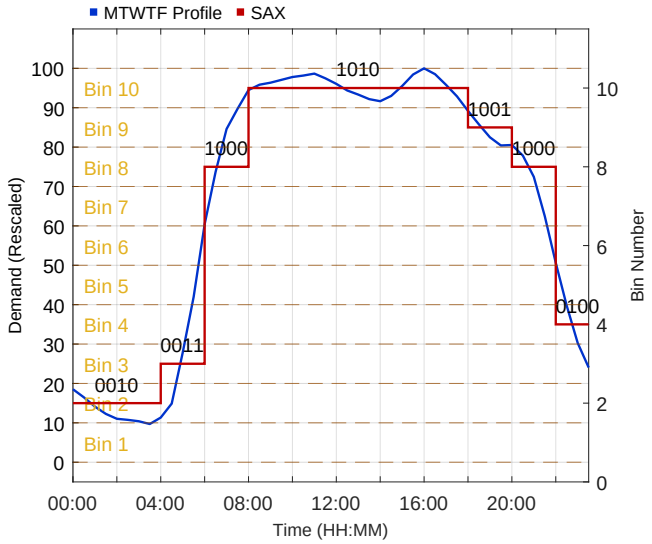


Figure 4.5 Weekday demand profile with SAX applied

In contrast to using techniques based on pattern sequence similarity, we extract singularities of bin data to create a series of lookup tables (LUT). Given the length of each piecewise segment, the process of creating a LUT for weekday, weekend day and month PAA or SAX representations are straightforward.

In this research, we present a LUT based on piecewise coefficient only. The main advantage of using the PAA approach in this context is that it requires less computational effort when compared to symbol mapping techniques to achieve visualisation of the time series. Furthermore, segment centre points are placed at fixed, regular intervals which result in a cubic interpolation where many of the demand data characteristics are retained during the transformation. In other words, the higher the reduction ratio is, the worse the performance of calculated approximation. This combination of findings has important implications for developing an energy optimisation algorithm.

Given each PAA segment is equivalent to a 2 hr epoch, the time series original 245,424 data items are now reconstructed from just twelve elements for each day and month feature (Table 4.1). Using PAA opens the possibility to perform forecasting up to one calendar month based on weekday and weekend day LUT. Extending the time horizon further up to 12 months requires the month LUT. When a seasonal adjustment is included, forecasting beyond 12 months is achievable. The mathematical representation of seasonal adjustment is

derived using a straight line approximation of the 12-month moving average, i.e., $y = bx + a$ where $b = 0.000442$ and the y-intercept a is set to the initial calculated weekday value.

Table 4.1
Piecewise coefficient lookup table

Name	Parameter
Weekday	[21.00, 10.47, 24.00, 77.11, 95.94, 98.02, 93.98, 94.64, 96.79, 84.46, 73.32, 21.00]
Weekend day	[21.00, 3.80, 3.29, 29.24, 55.42, 60.76, 53.30, 51.31, 59.67, 58.02, 55.84, 21.00]
Month	[40.11, 32.81, 30.23, 29.39, 29.00 34.97, 44.18, 57.63, 61.01, 65.00, 63.33, 53.23]

4.5 Piecewise interpolation

When reducing the dimensionality of extensive data using PAA, a compromise must be reached between how much the dimensionality of the original data can be reduced and the capacity to maintain competitive results. Cubic interpolation is used to obtain a somewhat smoother interpretation of the graph first created using the piecewise coefficient lookup table. Calculating a cubic polynomial that interpolates points of interest helps restore the shape of the original demand forecast profile. The centre point of each PAA segment defines a set of evenly spaced nodes. The piecewise function $S(x)$ interpolates all local data points and hence confines the ill-effects of any erroneous data points, Equation (4.5).

$$S_i(x) = a_i + b_i(x - i_{lo}) + c_i(x - i_{lo})^2 + d_i(x - i_{lo})^3 \tag{4.5}$$

Where $i \in [0, 1, \ldots, n]; x \in [lo, hi]$; where lo and hi define the start and end data points of each PAA segment, respectively (see Appendix D for worked example). The cubic polynomial coefficients are represented by the parameters a_i, b_i, c_i and d_i (Table 4.2). Tuning the first and end polynomial interpolants helps prevent extreme endpoint behaviour and improves concatenation of weekday and weekend day demand profiles. A $13 \times 4 \times 2$ multi-dimensional array defines a new polynomial coefficient structure for weekday and weekend day (Table 4.2).

Table 4.2
Piecewise cubic polynomial coefficient lookup table

Weekday				Weekend Day			
a_i	b_i	c_i	d_i	a_i	b_i	c_i	d_i
21.000	0	0.512	-0.256	21.000	0	0.750	-0.375
21.000	-1.024	-1.024	0.155	21.000	-1.501	-1.501	0.200
10.470	-1.755	0.841	0.111	3.802	-3.895	0.902	0.010
24.002	10.294	2.171	-0.256	3.296	3.801	1.022	-0.088
77.116	10.563	-2.104	0.160	29.242	7.771	-0.029	-0.069
95.942	1.409	-0.185	-0.009	55.425	4.212	-0.861	0.035
98.022	-0.518	-0.297	0.044	60.764	-0.976	-0.436	0.053
93.986	-0.802	0.226	0.004	53.302	-1.901	0.205	0.037
94.648	1.196	0.273	-0.109	51.312	1.490	0.643	-0.123
96.800	-1.872	-1.041	0.184	59.670	0.717	-0.836	0.138
84.466	-1.344	1.173	-0.383	58.022	0.675	0.826	-0.283
73.323	-10.360	-3.427	0.687	55.848	-6.283	-2.565	0.490
21.000	-4.814	4.814	-1.203	21.000	-3.309	3.309	-0.827

Figure 4.6 shows the result of applying cubic spline interpolation, using Equation (4.5) and the coefficient values listed in Table 4.2. The small blue circles mark the centre point of each of the 12 PAA segments, which yields 13 interpolation line segments. A similar plot is created for weekend days.

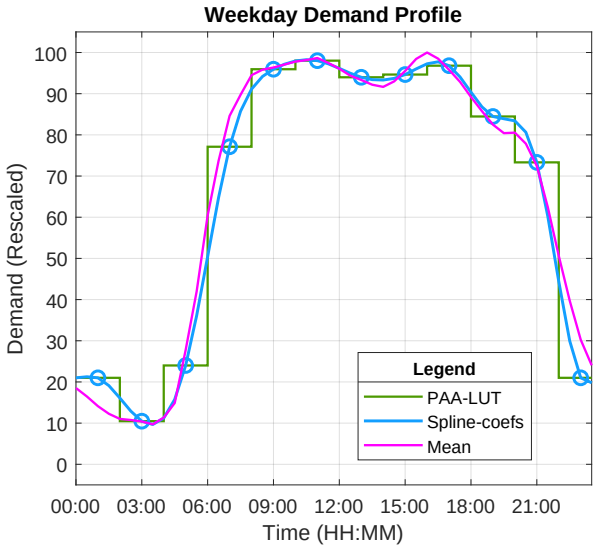


Figure 4.6 Weekday demand profile with cubic spline interpolation applied

4.6 Demand forecast function

A key problem is addressed by following the three-stage approach, that is: (1) data analysis, (2) reduction, and (3) smoothing. Equation (4.5) returns a normalised demand value. Given 12 PAA segments, this yields 13 start (lo) and end (hi) data points over a 24 hr period, i.e., $\tau_{(n)}(lo_n, hi_n)$, where $n \in \{0, 1, \ldots, 13\}$, and $\tau_{(1)}(lo_1, hi_1) = \tau_{(1)}(0, 2), \tau_{(n+1)}(4n-2, 4n+2), \ldots, \tau_{(N)}(4N-2, 4N)$. The second challenge is to map the start and endpoints to a time. The function then becomes useful because it can return a time-specific demand value.

Consequently, given any time and date (e.g., Monday 20th January 2020 14:15), calculating the demand value is relatively straightforward. Furthermore, the approach allows us to predict demand values over a finite horizon window simply by running the demand function by repeatedly incrementing the time by a desirable time interval at each iteration. Algorithm 1 shows the pseudocode for the demand forecast function. A full code listing and description are provided at Appendix D.

Algorithm 1 Demand forecast function

inputs:

date_time ← `date time` ▷ `eee dd-MM-yyyy HH:mm:ss`

outputs:

dfv ▷ demand_forecast_value

find Monday prior to date_time
$h \leftarrow$ date_time (`HH`)
$m \leftarrow$ date_time (`mm`)
$x = (60h + m)/30$ ▷ time_idx
if $h = 0$ **then**
  $lo = 0$
else if $h = 1$ **then**
  $lo = 2$
else
  $lo = 2h - 2$
end if
get: number of days from Monday to date_time
get: month number ▷ April = 1
require: cubic polynomial coefficient LUT ▷ Table 4.2
set cubic spline interpolation polynomial coefficients
dfv $= a_i + b_i(x - lo) + c_i(x - lo)^2 + d_i(x - lo)^3$
adjust for seasonality ▷ Month LUT

4.7 Baseline and performance indices

Assessing the accuracy of the demand forecast is an important consideration. In reviewing the literature, Makridakis and Hibon [202] found that simple forecast methods do as well, or in many cases better than statistically sophisticated ones like ARIMA and ARARMA models. For information that contrasts the ARIMA model to the long-range and short range forecast provided by an ARARMA model, see Parzen [203]. Comparison of the findings with those from other studies confirms that the simplest benchmark in forecasting literature is calculated using the random walk. The forecast from a random walk model is equal to the last recorded observation. Thus the random walk model underpins Naïve2 forecasts. That is, $\hat{y}_{y+h|t} = y_t$, where $\hat{y}_{t+h|t}$ represents the estimate of y_{t+h} based on the data $y_1, \ldots, y_t$. Visual inspection of the demand time series shows the data contains daily, weekly, and monthly seasonal patterns Figure 4.3 and, if the dataset extends over years, a 12 month negative

secular trend with constant variability. Naïve2 forecasting model is well suited to seasonally adjusted data. Therefore, the first benchmark of the proposed methodology will be assessed using this method. In this analysis, we limit h to 7 days (336 samples) which incorporates the distinct variation between weekday and weekend day seasonality. Thus, the forecast can be written as:

$$\hat{y}_{t+h|t}(t) = \begin{cases} y_{t+h}(t) \\ y_{t(h-7)}(t) \end{cases} \quad \text{where } h \text{ includes days of 1 wk (MTWFTSS)} \tag{4.6}$$

The second method used to compare the proposed methodology is based upon the simple notation for forecasts with a seasonal pattern $\hat{y}_{t+h|t} = (u_{t-1} + v_{t-1})s_{t-c}$, where c represents the weekly seasonality period index ($c = 336$), $\hat{y}_{t+h|t}$ is the h-step ahead forecast and,

$$\begin{aligned} \text{Level } u_t &= \alpha(y_t/s_{t-c}) + (1+\alpha)(u_{t-1} + v_{t-1}) \\ \text{Trend } v_t &= \beta(u_t - u_{t-1}) + (1+\beta)v_{t-1} \\ \text{Seasonality } s_t &= \gamma(y_t/u_t) + (1-\gamma)s_{t-c} \end{aligned} \tag{4.7}$$

where α, β and *gamma* are the smoothing parameters. The Holts-Winters additive method, Equation (4.7), is one of several exponential smoothing methods that can deal with seasonality and can be easily applied. However, for the Naïve2 and Holt-Winters forecasting models to remain effective, they are required to be re-trained as new observations become available. The lack of recent demand information for these models is a severe weakness and impacts the models continued performance.

In this work, four indices are used to evaluate the performance of individual forecasting progress. These include root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE) and the coefficient of determination or R Squared (R^2). A calculation that estimates the variance and differences using RMSE is defined as,

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(d_t - O_t)^2} \tag{4.8}$$

Where n denotes the number of observations, d_t are demand forecast predicted values and O_t are observed (actual) values at timestamp t.

MAPE is a measure that is widely used when comparing forecasting methods. The forecast error at time t is $e_t = O_t - d_t$. Hence, the percentage error $e_t = (O_t - d_t)/O_t$ so that the MAE for period t is,

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{O_t - d_t}{O_t}\right| \times 100 \tag{4.9}$$

MAE is a scale-independent parameter that is used to demonstrate the efficiency of the forecasting outcome.

$$MAE = \frac{\sum_{t-1}^{n}|d_t - O_t|}{n} \tag{4.10}$$

The coefficient of determination R^2 is derived using a ratio of explained variation ($SS_{regression}$) i.e., how well the regression model represents the actual demand data, to the total variation (SS_{total}), i.e., the variation in the observed data,

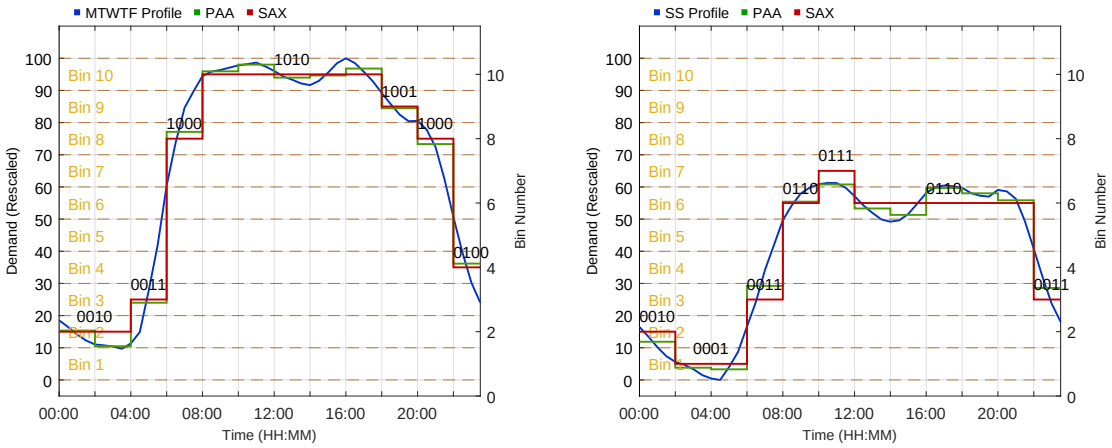
$$R^2 = \frac{SS_{regression}}{SS_{total}} \tag{4.11}$$

The process to analyse the prediction performances is described. Several benchmark tests are performed using a series of nominated test dates. For each specified test date, a new Holt-Winters estimation model is created using the previous four weeks of in-sample demand data. The forecast horizon window is set to include one complete week seasonal pattern, i.e., $h = 336$ ahead samples with smoothing parameters $\alpha = 0.82$, $\beta = 0$ and $\gamma = 0$. The construct of the proposed forecast model brings a distinct advantage for each forecast session, the practitioner can specify a start date and forecast horizon window. Therefore, the first set of tests compares the Holt-Winters benchmark model to forecasts generated using the same specified dates. Also, a single Naïve2 benchmark model created using in-sample demand data (27th June to 3rd July 2005) is compared to forecasts generated using the same nominated

test dates. The Naïve2 model functions on the same principle as the proposed forecast model, i.e., it is not immediately dependent on the availability of newly observed data.

4.8 Electricity demand forecasting

The methodology introduced in Chapter 4 has been applied to the UK electricity demand data (2005 to 2019). Figure 4.7 shows the following data over a 24 hr period: (1) enumerated mean demand data after dimensionality reduction technique (PAA) has been applied, (2) the 4-bit binary representation of the bin number that was assigned after symbolic discretisation (SAX), and (3) a plot of generalised demand data for weekday (MTWTF) and weekend days (SS).

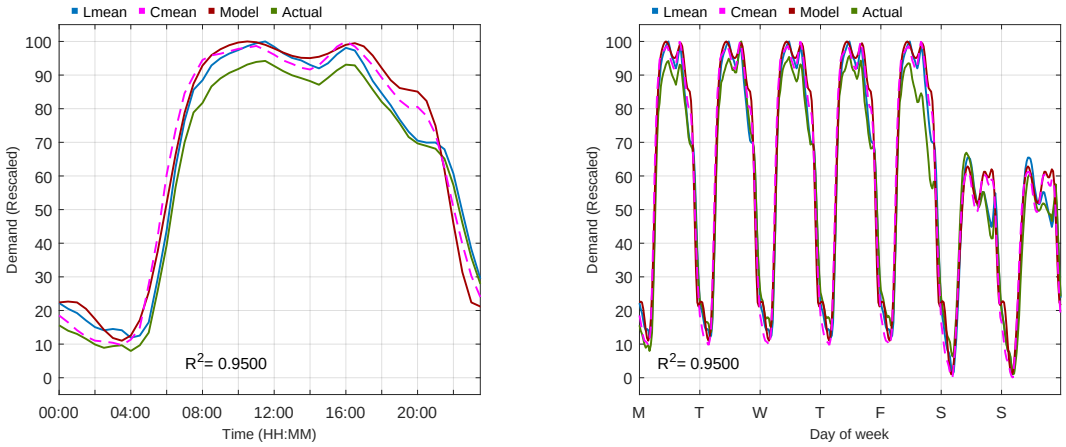


(a) Weekday **(b)** Weekend day

Figure 4.7 24 hr period PAA (2 hr segments) and SAX representations

It can be noticed that the effect of SAX encoding reduces the weekday and weekend day LUT further from 12 elements to seven. Although discretisation and SAX encoding offers the potential to reduce PAA dimensionality further, in the context of an energy optimisation system, a demand forecast based on PAA and piecewise interpolation have the potential to offer greater benefit. Results showing the cubic interpolants on the clamped discretised PAA subintervals are shown in Figure 4.8a. The plot compares the following four demand profiles: (1) actual demand data (Actual) measured over a 24 hr period on Monday 4th July 2005, (2) calculated cumulative mean value (Cmean) of 14 selected weekday demand profiles over 15 years (2005 to 2019), (3) calculated local mean value (Lmean) of four-weekday demand

profiles week commencing 4th July 2005, and (4) calculated demand data (Model) using the methodology described in Chapter 4. Figure 4.8b shows an extended seven day period which includes concatenated weekday and weekend day demand profiles. A measure how close the actual and model demand data over this seven day period is calculated $R^2 = 0.95$, $RMSE = 0.746$, and $MAE = 7.2262$.



(a) 24 hr **(b)** 7 d

Figure 4.8 Demand profile representations 4-Jul-2005

A summary of experimental results comparing forecast data against measured demand data and out-of-sample demand data are detailed in Table 4.3.

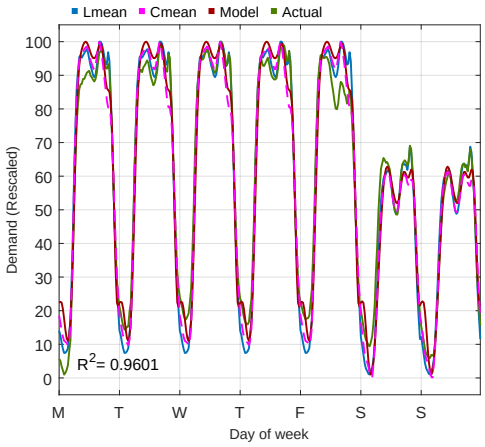
The performance of the demand forecast data shown achieves an average R^2 value greater than 0.92. The demand forecast and actual plot provide an excellent way to assess the goodness-of-fit of a regression at a glance. There is evidence the measure of performance is degrading slightly as time progresses. Figure 4.9a shows the demand profiles for week commencing 18th August 2014, and Figure 4.9b week commencing 5th August 2019. Nevertheless, these visual representations demonstrate weekday and weekend day recorded demand profiles (Actual) remain consistent with the model forecast data (Model). A generalised shape of the varying rates at which electricity is consumed during each 24 hr period is maintained.

Table 4.3
Performance of proposed model

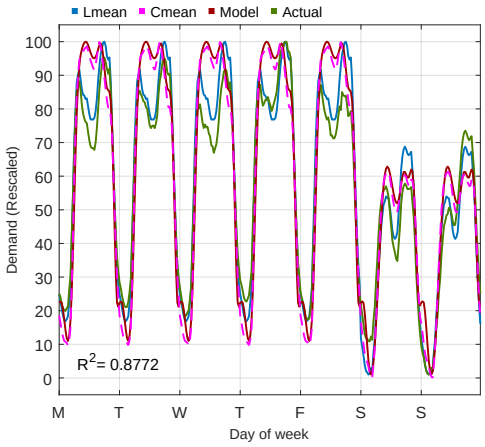
Date	$RMSE$	R^2
04-Jul-05	0.476	0.950
10-Jul-06	0.445	0.948
09-Jul-07	0.380	0.962
21-Jul-08	0.416	0.958
03-Aug-09	0.335	0.974
19-Jul-10	0.477	0.959
08-Aug-11	0.392	0.967
02-Jul-12	0.368	0.966
24-Jun-13	0.405	0.957
18-Aug-14	0.363	0.960
13-Jul-15	0.682	0.891
08-Aug-16	0.775	0.839
12-Jun-17	0.856	0.803
30-Jul-18	0.944	0.822
05-Aug-19	0.692	0.877
Average:	0.534	0.922

Table 4.4 reports the benchmark test results. Both MAE and MAPE values are presented when the forecast horizon $h = 336$ ahead. The figures show the out-of-sample Holt-Winters exponential smoothing forecasting accuracy is far more competitive than the proposed model, the MAE and MAPE average figures support this. This result is not unexpected and seems reasonable since the Holt-Winters model was re-baselined for each of the test dates. A visual comparison of Holt-Winters method and actual demand data for 18th August 2014 and 5th August 2019 are shown in Figure 4.9c and Figure 4.9d; a plot showing the model forecast for the same periods is added for reference. Further test results were derived comparing the second benchmark standard Naïve2 and actual demand data. The results of the Naïve2 model for within-week seasonality indicate the proposed model performance has a more significant benefit than the Naïve2 method.

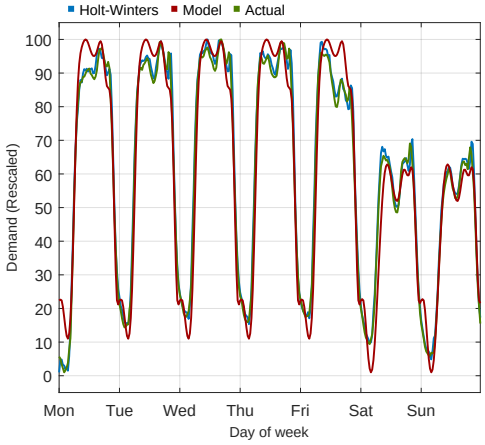
Figure 4.9e and Figure 4.9f show the Naïve2 method forecast against the actual demand data for $h = 336$ ahead periods commencing 18th August 2014 and 5th August 2019, respectively. For completeness, the proposed model forecast for the same period is shown. The corresponding MAPE figures confirm the relative performance of each of the models used. Predictably the Holt-Winters model outperforms the proposed model, which can be attributed to regular updates to the estimation data and relatively short forecast horizon



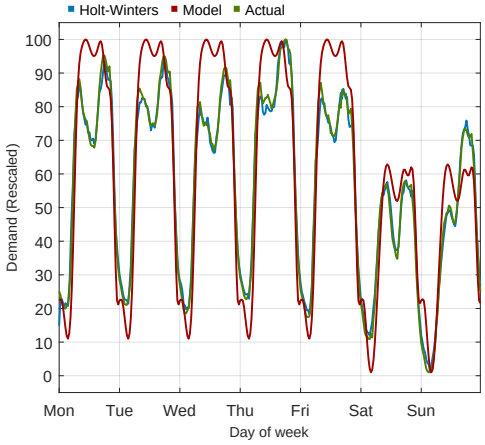
(a) Model 18-Aug-2015



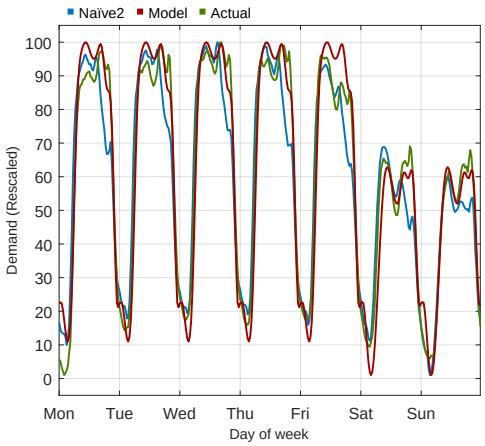
(b) Model 5-Aug-2019



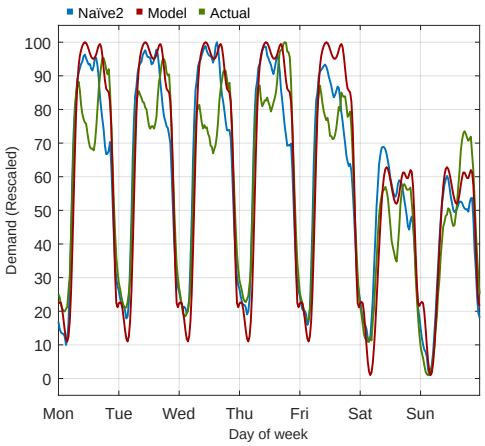
(c) Holt-Winters 18-Aug-2015



(d) Holt-Winters 5-Aug-2019



(e) Naïve2 18-Aug-2015



(f) Naïve2 5-Aug-2019

Figure 4.9 Demand profile representations $h = 336$ ahead

window. Figure 4.10 compares the MAPE figures derived from each model based on a single week ahead forecast. The relative performance ranking of the Naïve2, proposed

Table 4.4
Weekly MAE and MAPE (in%) on prediction of forecast horizon $h = 336$ ahead

Date	MAE			MAPE		
	Model	Holt-Winters	Naïve2	Model	Holt-Winters	Naïve2
04-Jul-05	7.226	1.270	2.930	19.330	3.230	8.860
10-Jul-06	6.016	0.690	10.150	15.490	2.470	30.610
09-Jul-07	5.355	0.670	10.780	14.330	2.470	33.760
21-Jul-08	5.765	1.200	10.260	16.630	3.960	33.430
03-Aug-09	4.642	1.520	9.970	15.280	5.310	35.770
19-Jul-10	6.805	1.000	10.440	17.550	3.860	34.890
08-Aug-11	5.882	1.490	11.070	23.450	5.900	48.160
02-Jul-12	5.419	0.820	10.780	16.240	2.180	41.880
24-Jun-13	6.168	1.070	10.110	16.350	3.000	31.840
18-Aug-14	5.223	2.050	11.440	28.120	6.080	44.590
13-Jul-15	9.440	1.350	13.180	24.330	3.650	49.060
08-Aug-16	11.123	1.740	14.070	38.960	4.900	46.190
12-Jun-17	13.027	1.870	14.270	36.370	4.760	38.690
30-Jul-18	12.895	1.990	17.230	57.060	9.560	84.800
05-Aug-19	10.412	1.970	13.730	53.040	7.010	41.180
Average:	7.693	1.380	11.361	26.169	4.556	40.247

model and Holt-Winters method is confirmed and consistent with earlier results shown in Table 4.4. While the proposed model overall performance figures are not equally comparable to the Holt-Winters results, it is reassuring the proposed model outperforms the widely used benchmark Naïve2 method. Furthermore, given the Holt-Winters model reliance to update the estimation data for continuous and affective forecasting and the proposed model ability to output short to medium term forecasts independent of any such updates, the proposed model will operate more effectively as part of a more comprehensive energy management system. It must be remembered that the proposed method is conceptualised for operation without any direct on-line measurement of the demand to be predicted, whereas the other methods require such measures.

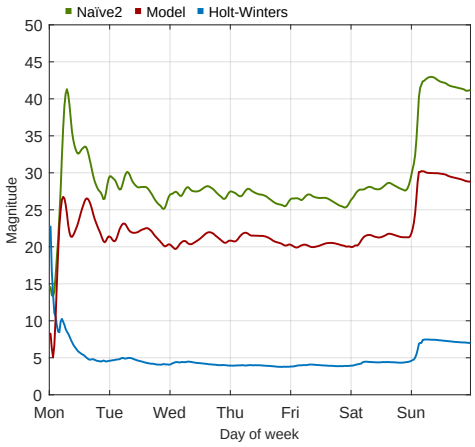


Figure 4.10 Comparison of MAPE results for $h = 336$ ahead commencing 5-Aug-2019

4.9 Summary

Knowledge of future electrical demand is essential for operators of community energy systems. The original contribution to knowledge put forward in this chapter is the methodology for calculating future electrical demand over a short horizon window. Machine learning based models designed for forecasting future energy needs are often opaque, difficult to interpret and require regular data interventions to ensure their usefulness [204]. However, the simplicity of this novel methodology means it can function without any direct on-line demand measurement or need to maintain an estimation dataset.

Using popular benchmark models, we have shown that despite the proposed model under performing when compared with a Holt-Winters seasonal model, the results outperform the seasonal naïve model forecasts. The demand forecast function (`demand.m`) is the improvement that further escalates the usefulness of the previous computational efforts. By applying simple mathematical reasoning, it is possible to establish a demand profile over a short horizon window. In the context of integrated demand response for community energy systems, these attributes may bring many benefits in comparison to more sophisticated approaches.

Chapter 5

Integrated Demand Response in an Energy System

5.1 Introduction

Technology innovation, guided by indicators, such as greenhouse gas emissions, is helping policymakers understand energy transition. Decarbonisation pathways are transforming ageing energy (electrical) infrastructures into more flexible decentralised systems. Demand response provides energy flexibility, which can improve network resilience and stability of operations. There is growing interest in advancing DR as the momentum for transitions to low-carbon, decentralised power systems become more mainstream. Studies show thermal inertia means community buildings have an essential role in demand response. However, although there is a growing amount of research about smart cities, there have been few investigations into the impacts of similar technology insertions in more remote community power systems.

To help fill this gap, this chapter describes the technical development of integrating demand response services in a community energy system. The main contribution of this of work is the implementation of a modified Dijkstra's algorithm, developed to optimise energy consumption of a heating and cooling system. This is achieved by setting the temperature setpoint value on a trajectory that follows the shortest-path between the current measured temperature and predicted temperature setpoint that updates at 10 min intervals over a short horizon window (nominally 4-h). The temperature setpoint trajectory (the optimal path) is influenced by the following principle parameters:

- Electrical demand consumption
- Cost (tariff)

- Occupant perceived thermal comfort

The implementation consists of a simplified lumped model for electrical demand forecasting, a scheduling subsystem that optimises the utility of energy storage assets, and an active/pro-active control subsystem. The active control strategy provides secondary demand response services, through optimising a multi-objective cost function formulated using a weight-based routing algorithm. In this context, the total weight of each edge between any two consecutive nodes is calculated as a function of thermal comfort, cost (tariff) and the rate at which electricity is consumed over a short future time horizon. The pro-active control strategy provides primary DR services. Furthermore, tertiary DR services can be processed to initiate a sequence of operations that enables the continuity of related electrical services for the duration of the demand side event. Later, in subsequent chapters, experimental studies will demonstrate the real-time operation of the proposed system on a prototype platform.

5.2 Generic framework

A generic decentralised optimisation and control framework can be used as part of an evolving demand response service; this means both curtailment and generation. This general arrangement will support primary and secondary DR services through frequency regulation and optimal control mechanisms, respectively, and tertiary DR events (Figure 5.1). Here, optimal performance might be described in terms of energy cost, thermal comfort and predicted future energy demands. A multi-objective cost function formulated using a weight-based routing algorithm automatically regulates the control of heating to create a meaningful energy demand reduction by shifting energy consumption to out of peak demand periods. Thermostatically controlled loads (TCL) can provide auxiliary services [205]. In this approach, the proposed scheme offers a pro-active control mechanism that changes the TCL operating setpoint proportionally to measured grid frequency. Following this approach avoids synchronisation problems that bound the coupling between frequency excursions and load dynamics that switch when prescribed frequency thresholds are exceeded [54]. An optimisation algorithm that responds to the real thermal needs of the building occupants is proposed. To achieve this, individual occupants can report their thermal comfort needs using

smartphone technology. The feedback reports are processed, and a consensus determined, which in turn is used to influence the room temperature.

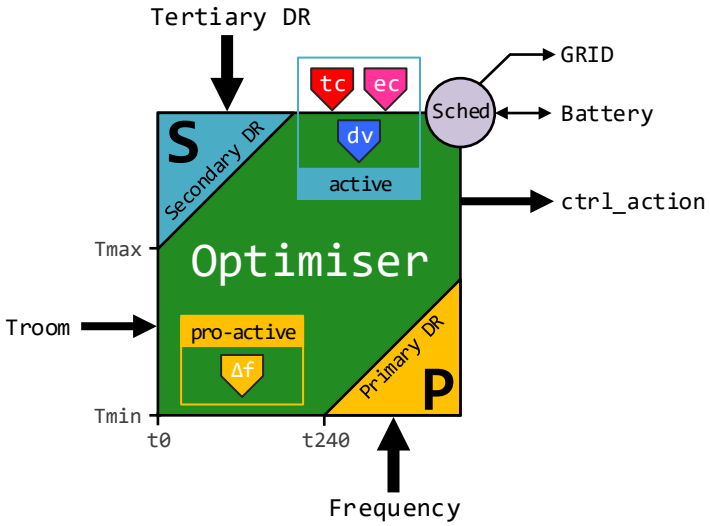


Figure 5.1 A demand response framework block diagram

The inclusion of building occupant feedback is crucial. Recent research has illustrated that engineers tend to assume occupants will not feel small changes in temperature [58]. This oversight can cause a performance gap between the expected and actual results from technologies intended to reduce or shift energy consumption in buildings. The inclusion of occupant feedback ensures that this issue will be avoided in the case of the solution presented in this research.

This chapter provides a reference basis for further DR applications in decentralised community based environments. It is particularly relevant to microgrids that are isolated from the grid as it offers potential for reducing the amount of energy storage required to balance the power fluctuation on those isolated microgrids. Current research has shown that even in the case of a single consumer, a microgrid option could be more economical than network renovation (e.g., provision of underground cabling) to increase the reliability [206]. Therefore, the ability to reduce the costs further by utilising the approach described in this chapter could offer real potential for the development of islanded and semi-islanded microgrids in many contexts.

5.3 General description

The proposed decentralised, informatics, optimisation and control simulation model has been developed to optimise space heating, schedule utility of energy storage assets and provide pro-active/active control for primary and secondary DR services. Two groups define the simulation model data so that system configuration parameters can be differentiated from local preferences. Ultimately, the simulation model is designed to assess our understanding of the optimiser and control components in the context of decentralised energy management. The applicability of the optimiser and control component is further demonstrated in hardware-in-the-loop (HIL) simulation.

The following outline is provided as an overview of the proposed optimisation and control strategy. The approach is based on the idea that when the demand for electricity on the distribution network is high, then the system attempts to reduce the local rate of energy consumption by reducing the space heating temperature setpoint. Similarly, during periods of low electricity demand the constraints that govern the temperature setpoint are relaxed, which in turn, allows, not mandates, an increase in energy consumption by increasing the space heating temperature setpoint.

When we add a measured response from occupants that describes their collective relative thermal comfort, the perception is the rate of energy consumption shifts towards being self-regulatory. For example, if the demand for electricity increases, the system attempts to reduce the local energy consumption at a rate that is inversely proportional to the predicted demand. If space remains void of occupants, this approach is satisfactory and local settings ensure a minimum space temperature is maintained. However, during periods of occupancy, individuals become eligible participants in the optimisation algorithm. Subsequently, when individuals report they are feeling cold, and their collective measured response satisfies a set threshold, then the resultant action is to issue a command that counters the instruction to reduce the space temperature further. Conversely, this self-regulatory behaviour works equally well during periods of low demand. Consider now introducing a third data type. Incentivising energy reduction through financial gain aims to reduce or shift energy consumption during periods of high demand [207].

Including information about the cost of energy into the mix introduces an interesting dynamic to the optimisation and control strategy. Given a time of use tariff that increases at times when demand is known to peak, the net contribution to the optimiser is to automatically adjust the energy consumption when the cost of electricity exceeds a user-defined threshold. Furthermore, the system can be configured to automatically switch to an alternative power source if demand exceeds a set limit or during periods when the cost of energy makes utilising an alternative power source more attractive (e.g., energy storage assets).

The immediate outcome attributed to the interaction between the three data types becomes even more attractive if their behaviours can be predicted over a finite time horizon. The opportunity to participate in tertiary DR services by making ready the system in response to a network operator DR instruction becomes feasible. The proposed control algorithm alters the demand profile trajectory such that it adds bias to the tri-data mix in a way that promotes a rise in space temperature. The net effect is to provide optimal space pre-heating in advance to commencing the scheduled DR event. Furthermore, a switching mechanism denies use of a local energy storage asset for a period leading up to the DR event. Instead, resources ensure the energy storage asset is set to recharge. Thus, when the DR event period commences the system power source automatically switches to the energy storage asset. Previous interventions ensure the energy storage asset capacity is sufficiently charged to enable it to remain the primary source for the duration of the event or until the asset can no longer meet the power demand for continued operation. In this instance, the grid becomes the systems primary power source, and recharging of the energy storage asset is initiated.

The remainder of this chapter describes the technical development of individual systems that contribute to the optimisation and control framework. Real-time computer simulations that aim to model the behaviour of physical systems and the mathematical model of the proposed optimisation and control algorithms are performed using the MATLAB/Simulink® environment. Level-2 MATLAB System functions have been used extensively during the design and implementation, providing access to create custom blocks that support multiple input and output ports. Furthermore, this section describes how desktop simulations are reconfigured to validate the optimisation and control algorithm using HIL simulation techniques.

The desktop simulation model is shown in Figure 5.2. In addition to the optimise and control block, the model is composed of a catalogue of supporting subsystems: energy, building, scheduler, date-time (dt) and demand event signal (des). The design and operation of the energy and building subsystems, which were introduced in earlier chapters, have been elaborated further in Appendix D. The remaining four subsystems have a more prominent role in energy optimisation, thus each subsystem contribution is discussed in the following sections.

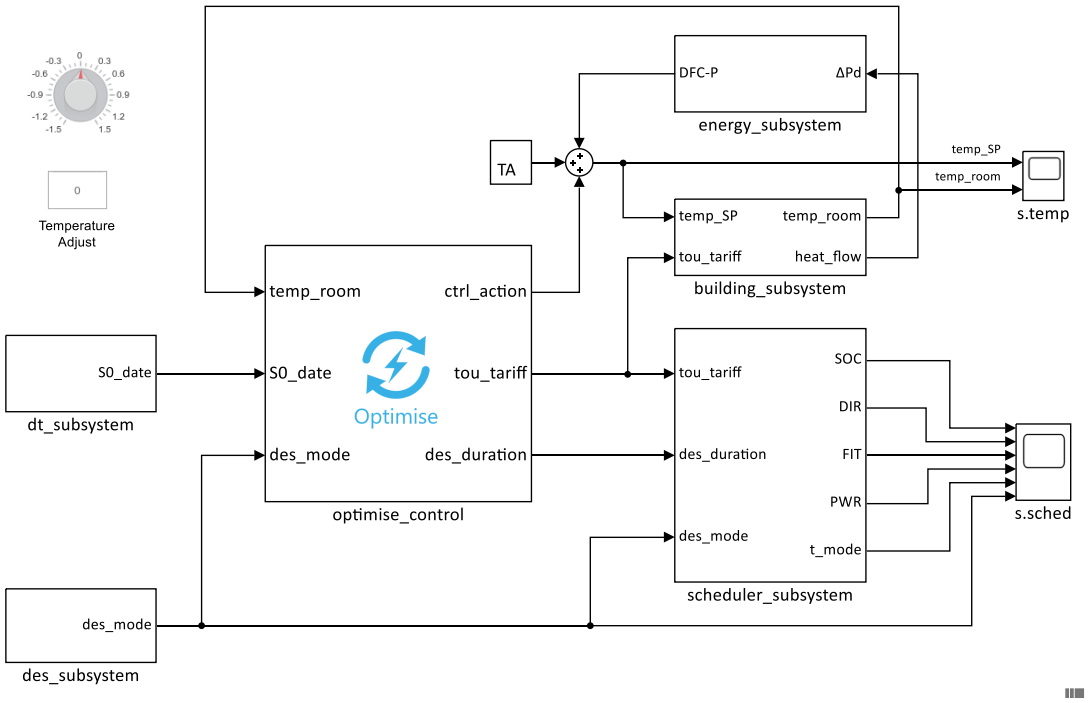


Figure 5.2 Simulink® model of energy optimisation framework

5.4 Technical development

The simulation optimiser is constructed in a piecemeal fashion, progressing sequentially by solving problems centred on three data types: (1) thermal comfort, (2) electricity demand forecast, and (3) cost (tariff). In brief, during periods when the system is not responding to a tertiary DR activity, the process begins by calculating a predicted or actual value for each data type over a 4 hr horizon window at 10 min intervals. Values are mapped onto a multi-dimensional array with a fixed number of rows (magnitude) and columns (time). A Dijkstra's algorithm is then used to project the predicted values over the 4 hr horizon

window [208, 209]. The contribution of each data type is then combined before k-means clustering (see, [210]) is applied iteratively at each 10 min interval. The result yields a new path that follows the optimal temperature setpoint trajectory over the 4 hr horizon window. For demand response applications, a model for building design can be successfully implemented using a simplified FOPDT model [189]. Time constants of 10 to 30 min and dead-times between 0 to 5 min are typical [211]. Avoiding complex calculations is achieved by taking a pragmatic approach when determining model control actions. For example, the proposed optimiser has been configured to update the control action at a sample time 10 min.

Since the control objective is to minimise the deviations from a temperature setpoint, according to the system and user-defined rules, at discrete points in time, the optimal cost (shortest path) can be obtained by formulating a Dynamic Programming algorithm that proceeds backwards in time. The algorithm takes a sequence of k-means centroid points, where each centroid represents a value that minimises the total intra-cluster variance of all objects in each cluster. In simple terms, given a time horizon of 240 min, this equates to 24 stages, each separated by a 10 min interval. At each stage, there are 11 objects. A k-means algorithm is applied to find the centroid of the 11 objects, at each stage. These calculations result in a series of 24 centroids that contribute to formulating the shortest path.

The objects that belong to each cluster are derived from a series of functions that calculate occupants' relative thermal comfort cost (tc), rate of energy consumption (demand forecast value) cost (dv), and energy cost (ec). Given the deterministic problem can be formulated in a bounded operating environment G, which can be equivalently represented by a gridmap of fixed dimension, the problem starts from a source node κ_s where $\kappa_s = \kappa_0 = G_{(j,S_0)}$, proceed to $\kappa_1 \in S_1$ and progresses to the final node $\kappa_t = \kappa_n = G_{(j,S_n)}$. An important characteristic of this activity is highlighted. In solving the shortest path problem, the source node κ_s and target node κ_t are revealed to the optimiser just before the first transition from $S_0 \mapsto S_1$ begins. The trajectory of the shortest path from $S_0 \mapsto S_1$ will follow a series of weighted edges η that interconnect successive pairs of nodes, i.e., $(\kappa_0, \kappa_1), (\kappa_1, \kappa_2), \ldots, (\kappa_{n-1}, \kappa_n)$.

In the framework of the fundamental problem, minimising the cost in a bounded operating environment G can be translated into mathematical terms:

$$J_n(i) = \min_{\kappa \in S_{n+1}} [c^n_{i\kappa}], \quad i \in S_n, \quad n = 0, 1, 2, \ldots, 24. \tag{5.1}$$

Where the cost of transition at $c^n_{i\kappa}$ is the centroid in a cluster of objects at stage S_n from node $i \in S_n$ to node $\kappa \in S_{n+1}$. For the problem to have a solution, each object centroid is constructed with a k-means algorithm. Here, after initially assigning a random object within a cluster as the first centroid, we compute the distance from each remaining object. Based on the square of these distances, a new centroid is defined. The process repeats until k centroids are chosen. We formulate the objects in the following paragraphs.

Also, when the network operator issues an explicit DR instruction, the optimiser initiates a pre-programmed control strategy that changes the trajectory of subsequent control actions in a period leading up to and during the event window. However, it remains useful if the control actions continue to respond to facility or occupant needs during this mode of operation.

5.5 Optimise and control subsystem

The optimise and control subsystem (`optimise_control`) is a user-defined block written using the MATLAB S-Function application programming interface (API). The proposed optimisation algorithm calculates the optimal space heating temperature according to the rate at which electricity is consumed (demand) and cost (tariff). Furthermore, the final temperature value is impacted by the occupants' thermal responses to the combined thermal effect of the environment and physiological variables that influence the relative thermal comfort.

Figure 5.2 shows the Simulink® optimise and control block includes three input signals: (1) room temperature (`temp_room`), (2) current date and time (`S0_date`), and (3) a demand event signal that indicates the status of a tertiary DR service (`des_mode`). The block output signals provide: (1) a control signal (`ctrl_action`) that will alter the space heating temperature setpoint, (2) the current cost of energy usage (`tou_tariff`), and (3) an indication of the tertiary DR event duration (`des_duration`). The internal architecture of the optimise and control subsystem is shown in Figure 5.3.

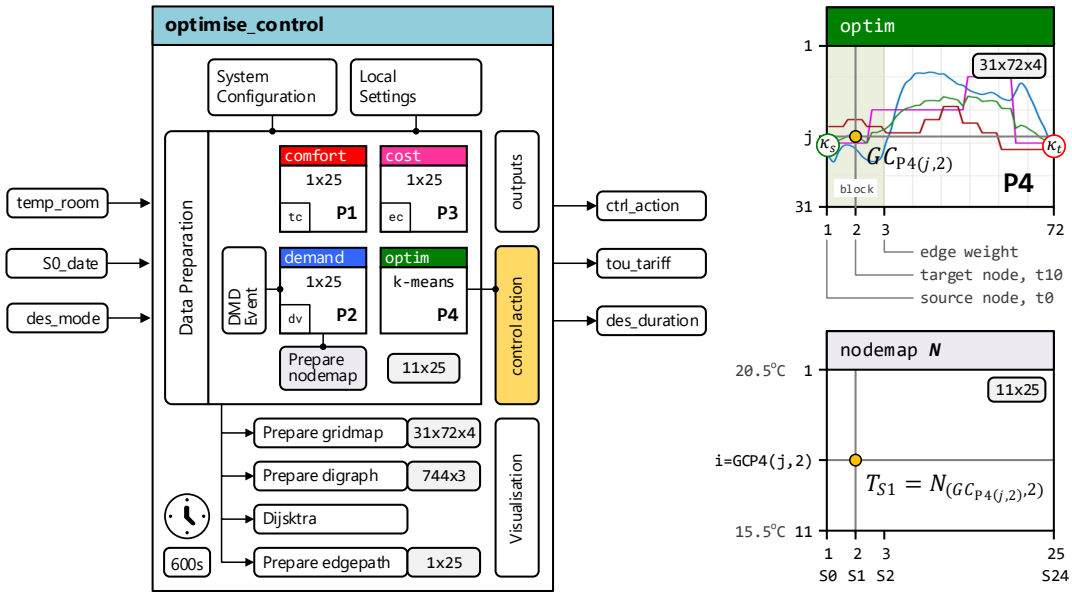


Figure 5.3 Optimise and control internal block diagram

The design presented in this article is configured to operate within a custom-built temperature range between $T_{min} = 15.5\,^{\circ}\text{C}$ and $T_{max} = 20.5\,^{\circ}\text{C}$. Exception handling ensures temperature values measured outside this range are mapped to 15.5 °C, or 20.5 °C. Default system configuration parameters set the forecast horizon window to 4 hr, a demand response temperature step (T_{step}) that instructs the control action to increase the space temperature by 2 °C over the duration of the forecast horizon window, and the duration of a demand event to 40 min. Additional system parameters specific to thermal comfort, electricity demand forecasting and cost (tariff) are described in the corresponding paragraphs that follow.

5.5.1 Thermal comfort

The energy demand of buildings is influenced by the presence and behavioural patterns of occupants [212]. The thermal comfort element impacts the temperature setpoint by analysing the measured room temperature (T_{room}) and occupants' feedback collated at a sample time of 10 min. Weekdays are divided into seven-time intervals $\tau_{(n)}$, configured to mirror a typical teaching timetable, whereas a weekend day consists of only one-time interval. Changing the weekend day interval pattern to replicate a weekday is straightforward. By considering occupant presence is inhomogeneous, for each $\tau_{(n)}$, we choose an algorithm for the simulation of occupants to be used as an input for current occupant level, u_k. In practice, not all

individuals will report their relative thermal comfort. Therefore the model automatically creates several feedback reports u_f, where $u_f \leq u_k$. An individual's response is measured using a unipolar Likert scale [213, 214]. The question has a five-scale response: too warm, warm, okay, cold, too cold; scored mathematically using a scale $u_f \in \{-2, -1, 0, 1, 2\}$. To imitate perceived behaviour patterns, for each time interval the following model parameters are defined: $u_{max} = \min u_f$, $u_{max} = \max u_f$ and response threshold u_{th} (%). The thermal model weekday parameters are reported in Table 5.1.

Table 5.1
Thermal model parameters

$\tau_{(n)}$	u_{min}	u_{max}
1	0	0
2	10	40
3	5	20
4	15	70
5	3	12
6	7	30
7	0	0

For any given weekday time, the thermal comfort model output is calculated by the following expression:

$$tc_{\tau_{(n)}} = \texttt{Mo}\left(\sum_{i=1}^{u_k} u_{f_{(i)}}\right), \quad u_f \in \{-2, -1, 0, 1, 2\}, \quad n = 1, 2, \ldots, 7; \tag{5.2}$$

with respect to:

$$\begin{aligned} \tau_{(1)} &= u_f(3:5); \\ \tau_{(2)(7)} &= u_f(2:5); \\ \tau_{(3)(6)} &= u_f(2:4); \\ \tau_{(4)} &= u_f; \\ \tau_{(5)} &= u_f(1:3); \end{aligned} \tag{5.3}$$

s.t. constraints:

$$(u_k/u_f) \times 100 > u_{th}; \tag{5.4}$$

$$u_{min} \leq u_k \leq u_{max} \tag{5.5}$$

For weekend days, we assume $u_f = 0$. Hence the model returns a value $tc_{\tau_{(1)}} = u_f(3)$. The variation in $\tau_{(n)}$ represents a bias that is configured to reflect a change in outside temperature over a 24 hr period.

It is noted the seven-time intervals $\tau_{(n)}$ are bounded by a start and stop clock time $\tau_{(n)}(t_1, t_2)$ such that $\tau_{(1)}(t_1, t_2) = \tau_{(1)}(00:00, 2n+7), \tau_{(n)}(2n+5, 2n+7)$; and terminating at $\tau_{(N)}(2N+5, 23:59)$. In practice, if a date and time are specified (e.g., `S0_date = Fri 05-Feb-2020 07:23:14`), then the task to determine if the date-time element occurs on a weekday or weekend day is straightforward. Given a date-time `S0_date` it is possible to formulate an algorithm that returns a 1×25 array $\delta_{tc} = [tc_0, tc_1, \ldots, tc_{24}]$ where tc_n represents a thermal comfort value over a 4 hr period at $10n$ min. It should be noted that because the optimiser is designed to take into consideration occupants' feedback in real-time at a sample time of 10 min $\delta(2:25) = tc_0 = tc_{\tau_{(1)}}$. However, if during the 4 hr horizon window the system identifies a time interval where $u_{max} = 0$, i.e., there are no planned occupants, the model starts a pre-programmed sequence that sets the thermal comfort on a downward trajectory reducing at a rate of 0.5 °C per 10 min interval until a minimum temperature threshold value T^{th}_{min} is reached. We have by the definition of the 11×25 nodemap δ_{tc} completed the data preparation of thermal comfort shown in Figure 5.3. It must be remembered that the thermal comfort model is formulated for operation within the simulated environment only. In practice, the implementation proposes occupants' report thermal comfort to the system using a smartphone app. This concept is elaborated further in Chapter 6.

5.5.2 Electricity demand forecasting

A data-driven methodology for modelling electricity demand forecasting is proposed [215]. The implication of this novel semi-autonomous simplified lumped model has the potential to offer decentralised electricity network operators' knowledge of the more extensive aggregated rate of future energy consumption. Thus, enabling decentralised energy management systems to proactively reduce load demand on small island electricity grids or distributed grid-edge systems as part of an evolving DR service. In this chapter, we integrate the electricity

demand forecasting model as part of the optimise and control framework. Initially, analysis of a chronological sequence of 245,424 discrete observations reveals the composition of the one-dimensional time series is characterised by three seasonal patterns: weekday, weekend day and month. These findings motivate an effort to reduce the dimensionality using piecewise aggregated approximation (PAA). Subsequently, calculating a cubic polynomial that interpolates points of interest yields a $13 \times 4 \times 2$ multi-dimensional array, which in turn helps restore the shape of the original demand forecast profile. The polynomial coefficient structure for weekday and weekend day are listed in the array page 1 and 2, respectively. Given both weekday and weekend day demand profiles recur every 24 hrs, it turns out using Equation (5.6) a normalised demand forecast value $M_i(x)$ can be tagged to a specific time in any 24 hr period.

$$M_i(x) = a_i + b_i(x - i_{lo}) + c_i(x - i_{lo})^2 + d_i(x - i_{lo})^3 \tag{5.6}$$

Where $i = 0, 1, \ldots, n; x \in [lo, hi]$, lo and hi correspond to the minimum and maximum data points of each PAA 2 hr segment respectively, and the cubic polynomial coefficient parameters are a_i, b_i, c_i, and d_i. Moreover, we shall show how the demand forecasting model can be used to compute a credible demand forecast value for any given date and time.

There are 12 equidistant segments, which equates to 13 periods (ρ) bounded by a minimum and maximum points lo and hi, i.e., $\rho_n(lo, hi)$ where the number of periods $n = 0, 1, ..., N$. In the first period $\rho_0(lo, hi) = \rho_0(0, 4n + 2)$, after that $\rho_n(4n - 2, 4n + 2)$; and terminating at $\rho_N(4N - 2, 4N)$. If we adopt the convention that makes 13-time intervals τ_n bounded by a start and stop clock time $\tau_n(t_1, t_2)$ then $\tau_0(t_1, t_2) = \tau_o(00:00, 2n + 1)$, after that $\tau_n(2n - 1, 2n + 1)$; and terminating at $\tau_N(2N - 1, 23:59)$. Thus, it can be seen, given a date-time `S0_date` it is possible to formulate an algorithm that returns a 1×25 array $\delta_{\mathtt{dv}} = [dv_0, dv_1, \ldots, dv_{24}]$ where dv_n represents a normalised demand forecast value over a 4 hr period at $t = 10n$ min starting from the specified date-time. This approach works equally well for both weekdays and weekend days.

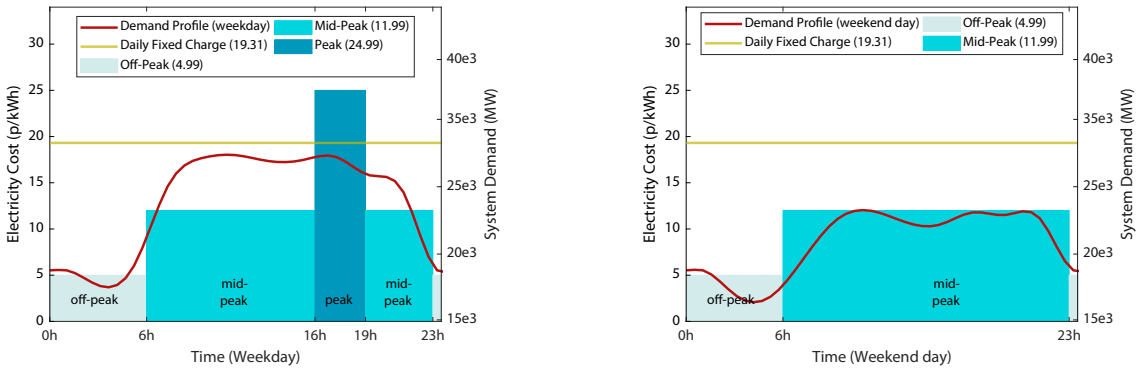
The normalised demand forecast value dv_n is defined as:

$$dv_n = N_{min} + \left(\frac{M_i(x) - DV_{min}}{DV_{max} - DV_{min}} \right) \times (N_{max} - N_{min}), \quad dv_n \in [1,11], \quad n = 0,1,\ldots,24 \quad (5.7)$$

where $N_{min} = \min_{m \in [n]} N_{(m,25)}, N_{max} = \max_{m \in [n]} N_{(m,25)}, DV_{min} = \min_{i \in [n]} M_i(x), DV_{max} = \max_{i \in [n]} M_i(x)$, noting that a nodemap n is a $m \times n$ two-dimensional array.

5.5.3 Cost (tariff)

A key consideration when taking part in a predefined energy reduction strategy must empower customers to use energy in the lowest price period accessible, at the same time as offering participation in DR initiatives. The cost (tariff) model is configured to integrate a typical static time of use (TOU) [216]. As shown in Figure 5.4, these tariffs charge cheaper rates when demand is low but increases for electricity consumption at peak times.



(a) Weekday **(b)** Weekend day

Figure 5.4 Model static TOU tariff

At a given date-time `S0_date`, the cost (tariff) simulation model returns a 1×25 array $\delta_{\texttt{ec}} = [ec_0, ec_1, \ldots, ec_{24}]$ where ec_n represents a normalised cost (tariff) value over a 4 hr period at $t = 10n$ min starting from the specific date-time.

The normalised cost (tariff) value is defined as:

$$ec_n = N_{min} + \left(\frac{EC(n) - EC_{min}}{EC_{max} - EC_{min}} \right) \times (N_{max} - N_{min}), \quad ec_n \in [3,9], \quad n = 0,1,\ldots,24 \quad (5.8)$$

Where $EC(n)$ is the cost (tariff) at $t = 10n$ min, $N_{min} = 3$, $N_{max} = 9$, $EC_{min} = 4.99$ and $EC_{max} = 24.99$. The scaling factors are set by design to position $\delta_{\texttt{ec}}$ values in the subsequent optimise stage such that a change in price to either off-peak or peak has maximum influence during the optimisation outcome. Furthermore, it will be shown $\delta_{\texttt{ec}}$ impacts the operation of system assets managed by the scheduler system.

5.5.4 Optimisation

The optimisation cycle Figure 5.3 starts on receipt of the input signal `S0_date`. Subsequent cycles commence at a block sample time of 10 min (600 sec). Previously, data preparation for occupants' thermal comfort, electricity demand forecast and cost (tariff) each returned a 1×25 array $\delta = [x_o, x_1, \ldots, x_{24}]$ where x_n represents a normalised data type (tc, dv and ec) value over a 4 hr period at $t = 10n$ min intervals starting from a specific date-time `S0_date`. Before each data type array can be processed, it must be homogenised in a way that makes it accessible to the optimiser. The data is transformed into a $m \times n$ two-dimensional nodemap $N(m,n)$ such that $\delta(x_n) \mapsto N(12-x, t_{10n})$. Accordingly, m represents a temperature $T = [T_{max} : -0.5 : T_{min}]$ and n defines 25 stages $(S_n \mid n \in \{0,1,\ldots,24\})$ each separated by a 10 min time interval for the duration of the 4 hr forecast window (e.g., $S_0 = t_0$ and S_{24} is linked to the 10 min time interval $t_{230} \mapsto t_{240}$). The 11×25 nodemap N is then transformed to a 31×72 gridmap G by the following function:

$$N(\delta(x_n), n) \mapsto G(i, 3n)_{\kappa_s}, G(j, 3n+1)_{\kappa_t}, \quad n = 1, 2, \ldots, 24; \tag{5.9}$$

where

$$i = 3\delta(x_n) - \Delta, \quad \Delta \in \{1, 2, 3\}; \tag{5.10}$$

s.t. constraints:

$$\Delta = \begin{cases} 1 & \texttt{if } \delta(x_{n+1}) > \delta(x_n); \\ 2 & \texttt{if } \delta(x_{n+1}) = \delta(x_n); \\ 3 & \texttt{if } \delta(x_{n+1}) < \delta(x_n) \end{cases} \tag{5.11}$$

$$j = \begin{cases} i+3 & \texttt{if } \Delta = 1; \\ i & \texttt{if } \Delta = 2; \\ i-3 & \texttt{if } \Delta = 3 \end{cases} \tag{5.12}$$

$$2 \leq \delta(x_n) \leq 10 \tag{5.13}$$

When the constraint is not satisfied, $\Delta = 2$.

The temperature from $t_0 \rightarrow t_{10} = T_{S_1}$, where $T_{S_1} \in \{T_{S_0}, T_{S_0} \pm 0.5\,^{\circ}\text{C}\}$ s.t. $T_{min} < T_{S_0} < T_{max}$, however if $T_{S_0} = T_{min}$ then $T_{S_1} \in \{T_{S_0}, T_{S_0} + 0.5\,^{\circ}\text{C}\}$. Furthermore if $T_{S_0} = T_{max}$ then $T_{S_1} \in \{T_{S_0}, T_{S_0} - 0.5\,^{\circ}\text{C}\}$. Based on this information, this equates to 31 permissible temperature changes between t_n and t_{n+10}. If we continue to record the change in temperature ΔT from $S_n \rightarrow S_{n+1}$ using blocks of three columns for each cycle, then it is clear a gridmap of size 31×72 is created. We refer to the three columns in each block as the source node κ_s, target node κ_t and edge weight $\lambda_\eta : \kappa_s \xrightarrow{\eta} \kappa_t$ respectively.

Dijkstra's algorithm computes the shortest path between a specified temperature point given at S_0 and S_{24}. This deterministic problem follows the principle of optimality, which suggests if the path taken transits from one legitimate node to the next minimises the cost-to-go from t_n to t_{n+10}, then the transition between the collective nodes must be optimal [6]. For the Dijkstra's algorithm to solve the shortest path, the 31×72 gridmap is first subjected to a series of simple transformations. The first instruction reshapes the gridmap into a 744×3 matrix referred to as the edgelist. Here, following the same convention to identify columns in the gridmap, the edgelist provides a listed description of all source nodes κ_n, legitimate target nodes κ_{n+1} and their respective connecting edge weights $\lambda_\eta : \kappa_n \xrightarrow{\eta} \kappa_{n+1}$, i.e., its associated cost. A second instruction creates a digraph object that generates an *Edges* variable (744×2 table) based on the number of source and target nodes extracted from the 744×3 edgelist, and a *Nodes* variable (275×1 table). The 275 value represents the total number of nodes κ_{275} in the fixed 11×25 nodemap. Finally, an equivalent sparse adjacency matrix representation of the digraph, which includes the edge weights, is created. Since the graph object we have constructed is a directed graph, the sparse adjacency matrix is not symmetric. However, we can overcome this by converting the sparse adjacency matrix

to a full storage matrix. In this instance, the conversion generates a 275×275 full storage matrix.

The data type shape is now in a format required by the Dijkstra's algorithm. Executing the Dijkstra's algorithm will compute the optimal cost which is equivalent to the summation of all edge weights $\lambda_n : \kappa_s \xrightarrow{\eta} \kappa_t$ on the shortest path from κ_s to κ_t between time t_0 and t_{240}.

The pseudocode describing the mathematical interpretation of the Dijkstra's algorithm is listed in Algorithm 2. Here we use w to represent a change in the edge weight that influences the calculation that solves the shortest path between each valid source vertex (u) and target vertex (v).

Algorithm 2 Dijkstra algorithm

```
for all w ∈ W do
    w ← weight(u,v)                          ▷ assign distance between each vertex
end for
for all v ∈ V-{s} do
    dist[v] ← ∞                ▷ initial distance from source to vertex v is set to infinite
end for
S ← 0
Q ← V
while Q ≠ 0 do                                                          ▷ main loop
    u ← minimumdist(Q,dist)
    S ← S ∪ u
    for all v ∈ adj[u] do
        if dist[v] > dist[u] + w(u,v) then
            d[v] ← d[u] + w(u,v)
        end if
    end for
end while
return dist
```

This process is repeated for each data type. At the end of each transformation the results are assigned to a specific page of a multi-dimensional array where page 1 (P1) is reserved for data type comfort, page 2 (P2) demand, and page 3 (P3) cost (tariff). The fourth page (P4) is reserved for the final stage in the optimisation process, which combines the contributions assigned to P1 to P3. Here, every third column in the 31×72 P4 gridmap is allocated a

grid centroid value $GC_{\texttt{P4}(j,s)} = 1$ where $j \in \{1,2,\ldots,31\}$ and $s \in \{3,6,\ldots,72\}$, and assigned to row index j that is equivalent to the k-means cluster centroid index that partitions the observations in the corresponding column s on P1 to P3. Note, for each data type $c^n_{i\kappa} = GC_{\texttt{P4}(j,s)}$, see Equation (5.1). The remaining values in each column are incremented by one until the row index j has reached its boundary limit, i.e., 1 or 31. When the Dijkstra's algorithm subsequently computes the shortest path between the source node $\kappa_s = GC_{\texttt{P4}(j,1)} = 1$ and the target node $\kappa_t = GC_{\texttt{P4}(j,71)}$ where $j = GC_{\texttt{P4}(j,72)} = 1$, the results yield the optimal path that transits from $S_0 \rightarrow S_{24}$. The control action $T_{S_1} = N(GC_{\texttt{P4}(j,2)}, 2)$. Simply stated, the control action is a fixed temperature value that is linked to the 11×25 nodemap $N(m,n)$ at row index $m = GC_{\texttt{P4}(j,2)}$ where $N(1,n) = 20.5\,^{\circ}\text{C}, N(2,n) = 20.0\,^{\circ}\text{C}, \ldots, N(11,n) = 15.5\,^{\circ}\text{C}$, where $n \in \{1,2,\ldots,25\}$. The relationship between the gridmap and nodemap is highlighted in Figure 5.3. The pseudocode describing the operating principle of the optimise and control algorithm is listed in Algorithm 3 (initialisation) and Algorithm 4 (main body).

Algorithm 3 Optimise and control: initialisation

inputs:

temp_room $= \{n | n$ is pos, and $15.5 \leq n \leq 20.5\}$ ▷ T_{room} (°C)

$S0$_date $\leftarrow$ `now()`

des_mode $= \{n | n$ is int, and $n \in \{0,1\}\}$ ▷ 0=normal, 1=event

outputs:

ctrl_action; tou_tariff; des_duration

initialise:

visual_mode; gridmap

horizon $= 4$ ▷ duration (h)

des_mode $= 0$

T_{step} (°C) $= \{n | n = 2$, and $n \in \{2,3\}\}$

des_duration $= \{n | n = 40$, and $n \in \{30,40,50\}\}$ ▷ duration (min)

$T_{min} = \{n | n = 16.0$, and $15.5 \leq n \leq 17.5\}$

$dt = \{$`tc, dv, ec, optim`$\}$

Algorithm 4 Optimise and control: main body

for every 10 min interval **do**
  $S0\_date \leftarrow S0\_date + 10$ min
  for each dt **do**
    if $dt =$ `tc` **then**
      $T_{min}^{th} \leftarrow T_{min}$ ▷ min temp threshold (°C)
      prepare comfort values $\forall Sn = \{n | n \text{ is an integer, and } 0 \leq n \leq 24\}$
    else if $dt =$ `dv` **then**
      require: des_mode; des_duration; T_{step}
      prepare demand values $\forall Sn = \{n | n \text{ is an int, and } 0 \leq n \leq 24\}$
      prepare node path
    else if $dt =$ `ec` **then**
      prepare tou values $\forall Sn = \{n | n \text{ is an int, and } 0 \leq n \leq 24\}$
    end if
    prepare gridmap
    adjacency matrix ← digraph ← edgelist ← gridmap
    optimise using Dijkstra's algorithm
    identify edgepath from start to end node $\forall Sn = \{n | n \text{ is an int, and } 0 \leq n \leq 24\}$
    if $dt =$ `optim` **then**
      prepare control action ▷ T_{S_1} (°C)
    end if
    get: visual mode
    display: visualisation ∈ {horizon,gridmap,bigpath,biggridmap}
  end for
end for

5.6 Demand event signal subsystem

The demand event signal subsystem (des_subsystem) simulates actions in response to a network operator instigated instruction. These signals are sent to individual customers enrolled in a campaign designed to deliver aggregated tertiary DR. The Simulink® model itself is trivial (Figure 5.5); however, the subsequent sequence of events requires further explanation. Firstly, the objective shifts to making the system ready for a DR event, this includes setting the control action to increase the room temperature in a measured approach by a preset value T_{step} (°C) within the 4 hr horizon window. Secondly, to ensure the battery energy storage system (BESS) is available with enough charge at the start of the DR event.

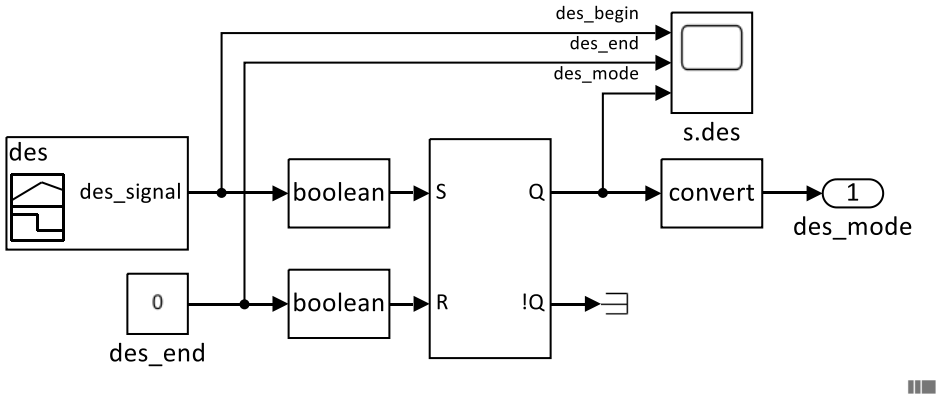


Figure 5.5 Simulink® model of demand event signal subsystem

The period of pre-heating is regulated by altering the demand forecast profile. By default, $T_{step} = 2\,^{\circ}\text{C}$. Therefore, the normalised demand forecast value $\delta_{\texttt{dv}} = [dv_0, dv_1, \ldots, dv_{24}]$ is recast to $\epsilon_{\texttt{dv}} = [dv_{\epsilon 0}, dv_{\epsilon 1}, \ldots, dv_{\epsilon 24}]$ where $\epsilon_{\texttt{dv}}(0:4) = dv_0$, $\epsilon_{\texttt{dv}}(i:j) = \epsilon_{\texttt{dv}}(i-5, j-5) - 1$ where $i \in \{5, 10, \ldots, 20\}, j \in \{9, 14, \ldots, 24\}$ s.t. $dv_0 - 1 \geq 1$. This new trajectory increases the last recorded room temperature by $2\,^{\circ}\text{C}$ at a rate of $0.5\,^{\circ}\text{C}$ every 50 min. At the beginning of each subsequent optimisation cycle, the trajectory leading up to the DR event is maintained, i.e., it advances closer to the plus $2\,^{\circ}\text{C}$ temperature at each iteration and towards the DR projected start time. However, before $\epsilon_{\texttt{dv}}$ reverts to $\delta_{\texttt{dv}}$, the trajectory is modified further, this time by reducing the temperature setpoint $2\,^{\circ}\text{C}$ less than the temperature recorded immediately before the start of the tertiary DR event. The system reinstates $\delta_{\texttt{dv}}$ immediately after the DR event terminates.

The des_mode signal triggers the scheduler subsystem to start charging the BESS. The energy storage asset will continue to charge until the start of the DR event. The battery will then start to work from this time, reducing the stored charge of the battery while it continues to provide primary power to the heating system. The heating system will continue to be supplied from the battery until a state of charge (SOC) minimum threshold has been reached. The scheduler switches primary power to the grid and the battery to charge.

5.7 Scheduler subsystem

The scheduler subsystem primary job is to monitor several signals and direct the operation of an automatic transfer switch between a grid and an alternative backup source of power. To ensure the appropriate power source is selected, the scheduler requires knowledge of the current cost (tariff) of electrical energy, whether a tertiary DR event is in progress including information of the event duration and BESS SOC. The Simulink® model of the scheduler subsystem is shown in Figure 5.6 and includes three input signals and six output signals. The output signals are provided for visual indication of various signal status. A simplified BESS element (ess_subsystem) simulates a battery SOC using a first-order transfer function. Locally defined parameters `SOC_hi` and `SOC_lo` set maximum and minimum SOC values (expressed as a percentage), which determine when the BESS is declared available for use. In this context, initial parameter values are defined as 80% and 20% respectively. The model also includes a self-discharge rate (SDR) which reduces the stored charge of the battery naturally over time.

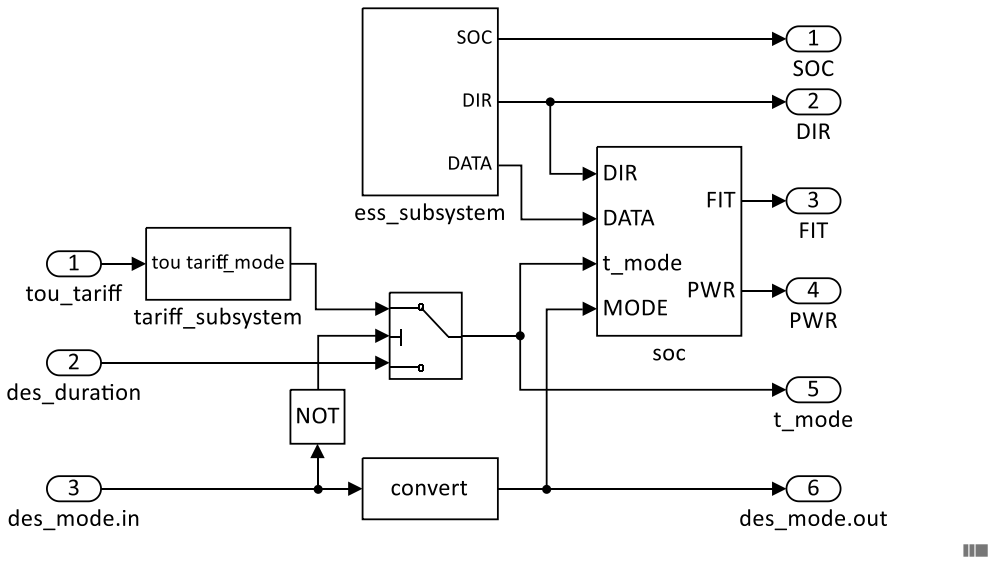


Figure 5.6 Simulink® model of scheduler subsystem

The BESS availability function is represented by Equation (5.14), where `SOC_lo` is a low-level SOC threshold (locally defined parameter).

$$\texttt{FIT} = \begin{cases} 0 & \texttt{if SOC} \leq \texttt{SOC\_lo} \\ 1 & \text{otherwise} \end{cases} \tag{5.14}$$

Control rules that determine when the primary power source is set to grid or BESS are illustrated in Figure 5.7. The decision variable `t_mode` is the cost (tariff) threshold and automatically switches the power source to BESS when the cost (tariff) is high s.t. Equation (5.14). Furthermore, when signal `des_mode=1` (0=normal, 1=tertiary DR event), `t_mode=0` thus preventing a control action that switches the power source to BESS during the period leading up to the start of the DR event (nominally 4 hr). Signal `CDir` reports if the battery is in charge or discharge (0=discharge, 1=charge); `PWR` denotes primary power source (0=grid, 1=BESS); `SOC_EC` denotes cost (tariff) in use, (0=TOU, 1=BESS).

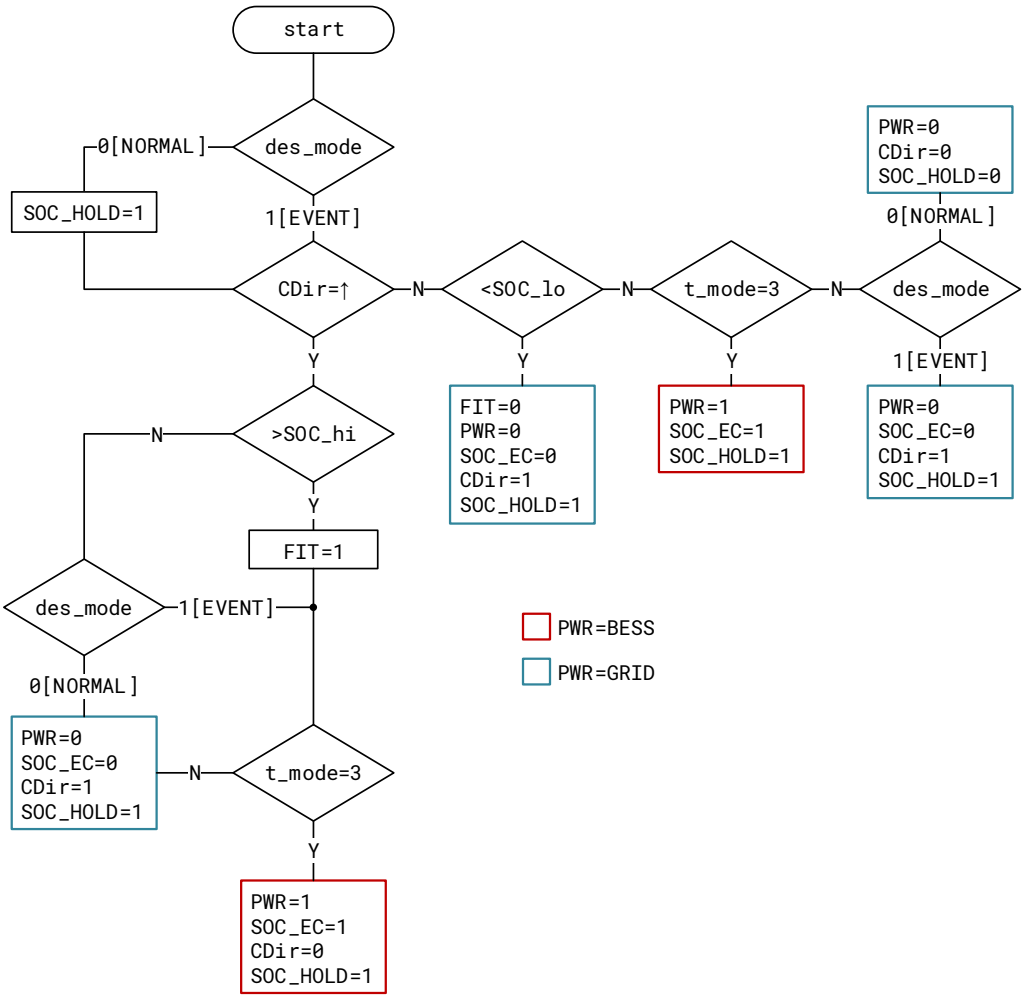


Figure 5.7 Scheduler control logic flowchart

The pseudocode describing the operating principle of the scheduler algorithm is listed in Algorithm 5.

Algorithm 5 Scheduler subsystem

inputs:

tou_tariff $= \{n | n$ is positive, and $n \in \{4.99, 11.99, 24, 99\}\}$ ▷ cost (GBP)

des_duration $= \{n | n = 40$, and $n \in \{30, 40, 50\}\}$ ▷ duration (min)

des_mode $= \{n | n$ is positive, and $n \in \{0, 1\}\}$ ▷ 0=normal, 1=event

outputs:

SOC; CDir; FIT; PWR; t_mode; des_mode.out

initialise:

t_mode $\in \{0, 1, 2, 3\}$; $ET_{th} \leftarrow 3$; SOC $= 0$; des_mode.in $= 1$

$\text{SOC}^{th}_{des} = 90\%$; $\text{SOC}^{th}_{max} = 80\%$; $\text{SOC}^{th}_{min} = 20\%$

for every 10 min interval **do**

t_mode←tou_tariff

require: SOC

if des_mode.in = 0 **then** ▷ normal

enable energy storage asset

if CDir detect increase **then** ▷ on-charge

if SOC $> \text{SOC}^{th}_{max}$ **then**

declare energy storage asset available for use ▷ `FIT=1`

if t_mode $= ET_{th}$ **then**

set power source to energy storage asset ▷ `PWR=1`

else

set power source to grid ▷ `PWR=0`

end if

else

set power source to grid

end if

else ▷ CDir detect decrease

if SOC $< \text{SOC}^{th}_{min}$ **then**

declare energy storage asset not available for use ▷ `FIT=0`

set power source to grid

else

if t_mode $= ET_{th}$ **then**

set power source to energy storage asset

else

set power source to grid

set energy storage asset self-discharge rate

end if

end if

end if

continued ...

Algorithm 6 Scheduler subsystem (continued)

```
    else                                                          ▷ demand event
        if CDir detect increase then                              ▷ on-charge
            if SOC > SOC_des^th then
                declare energy storage asset available for use
                if t_mode = ET_th then
                    set power source to energy storage asset
                else
                    set power source to grid
                end if
            else
                if t_mode = ET_th then
                    set power source to energy storage asset
                else
                    set power source to grid
                    enable energy storage asset self-discharge
                end if
            end if
        else                                                      ▷ on-discharge
            if SOC < SOC_min^th then
                declare energy storage asset not available for use
                set power source to grid
            else
                if t_mode = ET_th then
                    set power source to energy storage asset
                else
                    set power source to grid
                end if
            end if
        end if
    end if
end for
```

5.8 Date-time subsystem

The Simulink® model of the date-time subsystem (dt_subsystem) is shown in Figure 5.8. The primary function of the subsystem is to provide a date-time element at a sample time of 10 min. The model has been configured to run in real-time during experimental evaluation. By default, `dt` is set to the current date and time, using format `dd-mmm-yyyy hh:mm:ss`, with the option to set to any data-time during model analysis. The date time model parameters are reported in Table 5.2.

Table 5.2
Date-time system parameters

Parameter	Value
dt	`dd-mmm-yyyy hh:mm:ss`
C5	600
C6	1.157412771169e-5

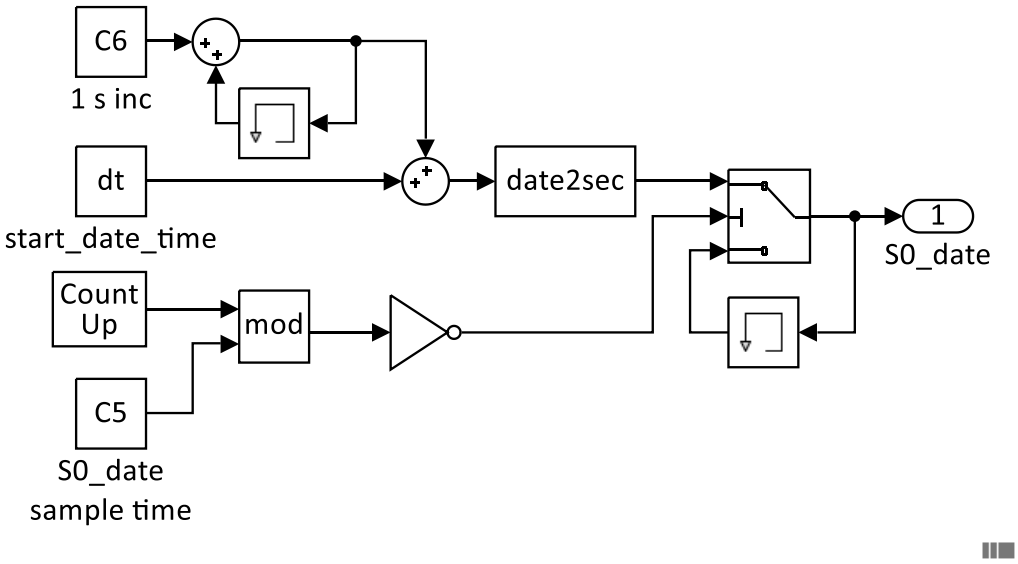


Figure 5.8 Simulink® model of date-time subsystem

5.9 Software code development

When considering the surrounding environment, it is useful to categorise factors that initiate change into three groups: (1) human factor, (2) technical means, and (3) external conditions.

In the context of energy management, human factor includes actors who make decisions based on experience, knowledge, and opinion in response to a situation. Their contribution is reflected in the behavioural change in the interconnected parts, transforming the output within a prescribed boundary. The technical means refers to those processes that operate autonomously during computational procedures and alter the decision-making processes that ultimately change the state of the system. Finally, the external conditions are monitored and trigger a change in system behaviour when predefined conditions are satisfied. External conditions remain intact, protected from any direct system change.

The computational study was carried out using the MATLAB/Simulink® environment. Figure 5.9 shows the computational study software code grouping. The `optim_ctrl.m` function is central to this diagram. The modified Dijkstra's algorithm has evolved to determine the shortest-path between current measured room temperature and predicted temperature setpoint (at 10 min intervals over a 4-h horizon window). The role of the modified algorithm is to optimise the energy consumption based on energy demand, tariff and user feedback of a short time horizon. A full description of MATLAB® programs (M-file) (except visualisation programs) are summarised in Table D.4. Code listing for each item, including the content of each binary MATLAB® file that stores workspace variables (MAT-file) are provided at Appendix D.

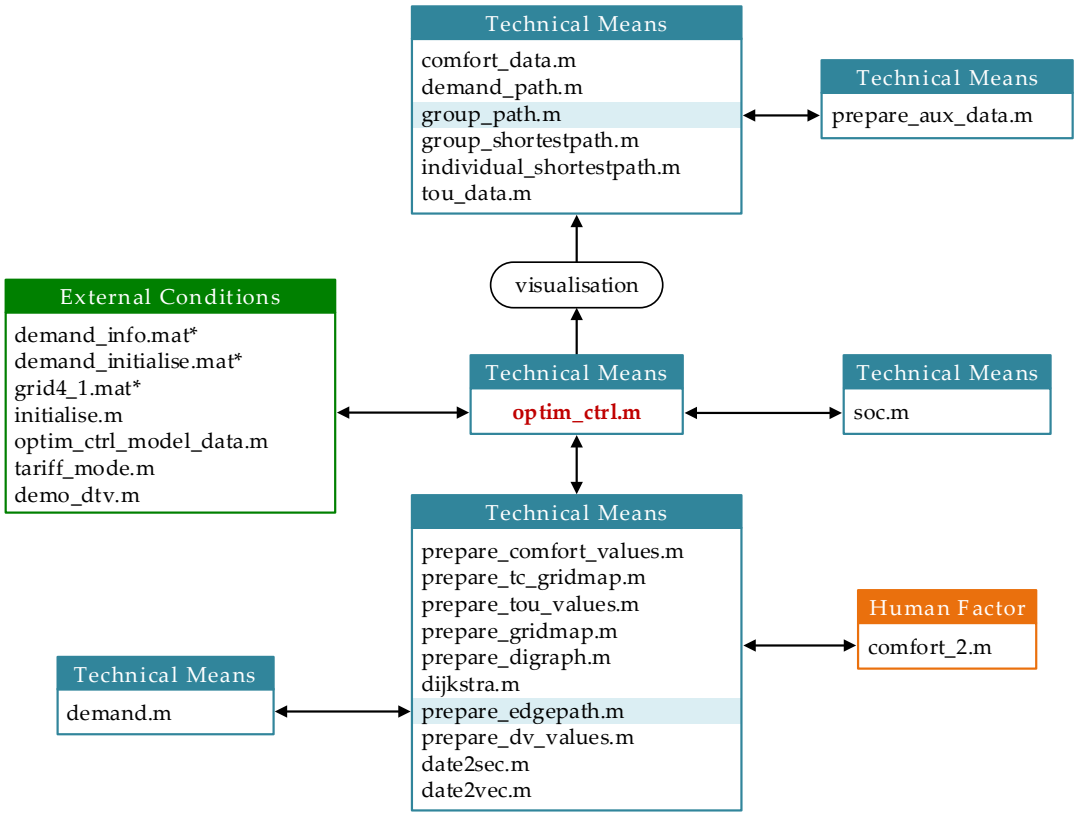


Figure 5.9 Software code groups

5.10 Computational study

In this section, we report the findings from a computational study (desktop simulation). By design, the computational study validates the functionality of critical services. In contrast, the experimental evaluation (Section 6.8) is explicitly directed on proving the interaction of proposed data types within the optimisation subsystem. The interaction between decision variables and control actions of individual subsystems is complex. Accordingly, the computational study validates the functionality of the following vital services:

- Thermal comfort model
- Electrical demand forecasting model
- Cost (tariff) model
- Optimiser
- Tertiary DR activity

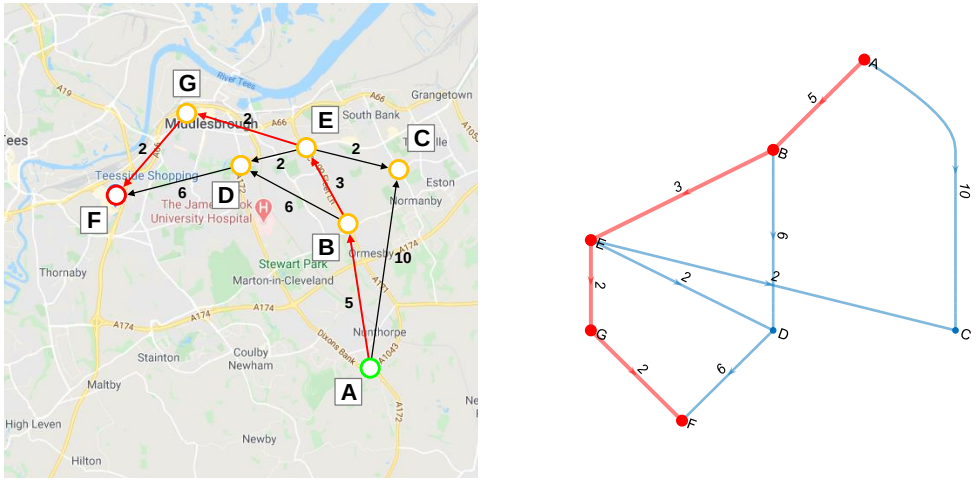
- Pro-active frequency control

A multi-objective cost function formulated using a weight-based routing algorithm is an essential component and plays a vital role in the energy management system. The importance of this working correctly justifies special attention. Therefore, we begin by reporting the test results obtained from the preliminary analysis and implementation of the optimisation algorithm.

5.10.1 Single source shortest path

As applications get complex and data-rich, a structured engineering design process demands meticulous attention to detail, coordination of data and a necessity to achieve the best performance. The energy management problem has been translated into a mathematical problem, which can be formulated using the dynamic programming approach. The energy management algorithms are motivated for overall optimisation of the problem, using the output of a set of smaller sub-problems to optimise a bigger problem. An adaptation of the single-source shortest path (SSSP) algorithm by Dijkstra is fundamental to the solution. Therefore, a familiar deterministic problem is chosen to evaluate the optimisation approach, ensuring the results are entirely predictable before introducing the software code to the energy management problem.

Figure 5.10a shows seven places of interest marked on a map. A unit distance and permissible direction of travel between each place (node) are shown. The problem is to find the shortest path between the nominated start node (A) and target node (F).



(a) Places of interest map **(b)** Gridmap

Figure 5.10 Shortest path problem

To solve the shortest path problem mathematically a weighted matrix table (WMT) is created (Figure 5.11). Starting from node A, the first row shows the distance between node A and each valid destination. Invalid paths are set to infinity. The boxed values indicate the shortest possible distance between a start and end node. After the shortest path has been declared the remaining column entries remain blank. The starting node in the proceeding row is the node with the shorted path declared in the previous row. If there are multiple valid nodes, we choose the most left shortest path entry in the table. If the path is found to be less than the previous entry, this new value is entered into the table followed by the letter that represents the start node for this leg. The rightmost column details the shortest path (distance and route) starting from node A. Solving the table identifies the shortest path between all valid start and end node combinations. The last row confirms the shortest path from node A to F is $A \rightarrow B \rightarrow E \rightarrow G \rightarrow F$. The total distant (cost) is 12 units.

	A	B	C	D	E	F	G	A to	
A	$\boxed{0A}$	$5A$	$10A$	∞	∞	∞	∞	0	
B		$\boxed{5A}$	$10A$	$11B$	$8B$	∞	∞	5	A→B
E			$10A$	$10E$	$\boxed{8B}$	∞	$10E$	8	A→B→E
C			$\boxed{10A}$	$10E$		∞	$10E$	10	A→C
D				$\boxed{10E}$		$16D$	$10E$	10	A→B→E→D
G						$12G$	$\boxed{10E}$	10	A→B→E→ G
F						$\boxed{12G}$		**12**	**A→B→E→G→F**

Figure 5.11 Weighted matrix table

Before the Dijkstra's algorithm software code can determine the shortest (optimal) path, the original route map is expressed in mathematical terms using a graph with directed edges (*digraph*).

$$s = [\ 1 \quad 1 \quad 2 \quad 2 \quad 4 \quad 5 \quad 5 \quad 5 \quad 7\ ]$$

$$t = [\ 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 3 \quad 4 \quad 7 \quad 6\ ]$$

$$w = [\ 5 \quad 10 \quad 6 \quad 3 \quad 6 \quad 2 \quad 2 \quad 2 \quad 2\ ]$$

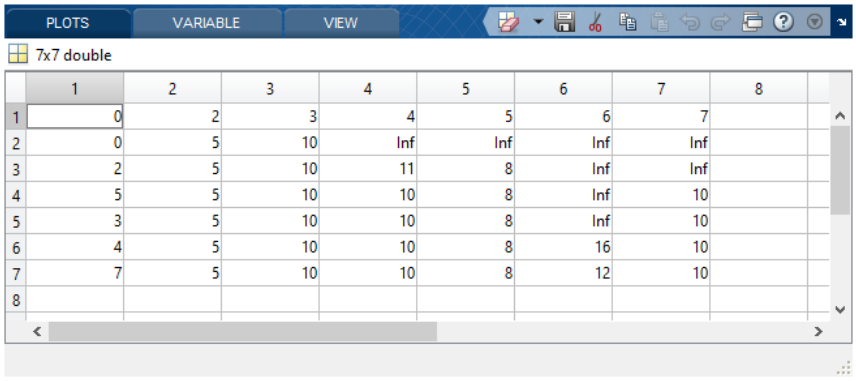
Given $A = 1$, $B = 2$ etc., and s and t are row vectors representing the start and target nodes of all valid paths (λ_k), and w is the associated distance (weight) from s to t, i.e., $\lambda_k = \kappa_s \xrightarrow{w} \kappa_t$. Such that $\lambda_1 = 1 \xrightarrow{5} 2$, $\lambda_2 = 1 \xrightarrow{10} 3$, $\lambda_3 = 2 \xrightarrow{6} 4$ and so on. The formal mathematical definition of this arrangement is an *edgelist*. The edgelist lends itself to matrix representation (Figure 5.12), where a full adjacency matrix $M = \lambda_k(m,n)$, where $\lambda_k(m \to n) = \lambda_k(\kappa_s \xrightarrow{w} \kappa_t)$ is the shortest path from κ_s to κ_t through $\{1, \cdots, k\}$.

		To						
	M	1	2	3	4	5	6	7
	1	0	5	10	0	0	0	0
	2	0	0	0	6	3	0	0
	3	0	0	0	0	0	0	0
From	4	0	0	0	0	0	6	0
	5	0	0	0	2	0	0	2
	6	0	0	0	0	0	0	0
	7	0	0	0	0	0	2	0

Figure 5.12 Full adjacency matrix

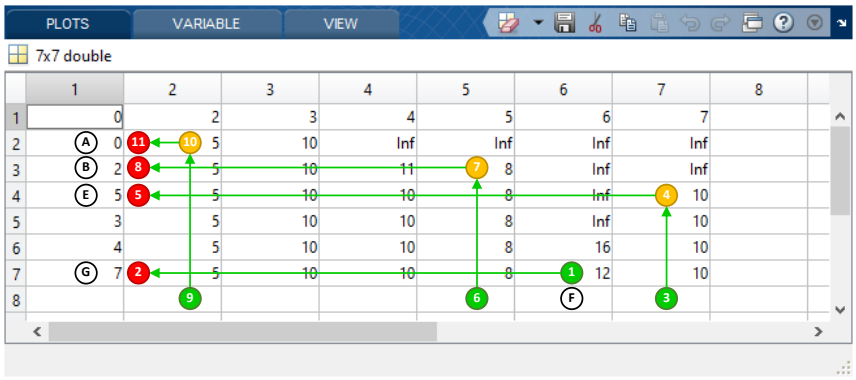
Referring to Figure 5.10a, the optimal path between node A(1) and F(6) is confirmed: $A \rightarrow B \rightarrow E \rightarrow G \rightarrow F$, and the total distance is 12 units. The pseudocode implementation of the Dijkstra's algorithm was introduced in Chapter 5, Algorithm 2. The software code function is defined as `[cost, path]=dijkstra(M,s,t)`, where M is the adjacent matrix, s $= \kappa_s$ and t $= \kappa_t$ (see Appendix D.11). The function first creates a weighted matrix table (see, Figure 5.13a) using the map description encoded in the adjacency matrix before returning the optimal route (path) and weight (cost) between the preset start and target nodes. For example, executing the function `[cost, path]=dijkstra(M,1,6)` returns cost=12 and path=[1, 2, 5, 7, 6].

To interpret the MATLAB generated weighted matrix table, row 1 `wmt(1,:)` reads [A B C D E F G] and column 1 `wmt(:,1)` reads [A A B E C D G]. The values in row 7 columns 2 to 7, i.e., `wmt(7,2:7)` represent the shortest path from node A(1) to each of the other nodes identified at row 2 columns 2 to 7. For example, the shortest path (cost) from node A(1) to node F(6) is `wmt(7,6)`, which is 12. Similarly the cost from node A(1) to node D(4) is `wmt(7,4)`, which is 10. The process to determine the path is more involved. Here we refer to Figure 5.13b. Starting at the destination node `F` then working backwards transiting through nodes until the starting node `A` is reached. In simple terms, from the destination node move in an upwards direction stopping *before* the next number in the same column changes, then move to the left most column, that is `F` $\rightarrow$ `1` $\rightarrow$ `2` $\rightarrow$ `G`. The digit in the left most column is



	1	2	3	4	5	6	7	8
1	0	2	3	4	5	6	7	
2	0	5	10	Inf	Inf	Inf	Inf	
3	2	5	10	11	8	Inf	Inf	
4	5	5	10	10	8	Inf	10	
5	3	5	10	10	8	Inf	10	
6	4	5	10	10	8	16	10	
7	7	5	10	10	8	12	10	
8								

(a) Computer generated weighted matrix table



	1	2	3	4	5	6	7	8
1	0	2	3	4	5	6	7	
2	A 0	11 10	5	10	Inf	Inf	Inf	Inf
3	B 2	8	5	10	11	7 8	Inf	Inf
4	E 5	5	5	10	10	8	Inf	4 10
5	3		5	10	10	8	Inf	10
6	4		5	10	10	8	16	10
7	G 7	2	5	10	10	8	1 12	10
8		9			6	F	3	

(b) Shortest path from node `A` to `F`

Figure 5.13 MATLAB implementation of Dijkstra's algorithm weighted matrix table

the column number where the next leg in the path starts, i.e., 7. Now, 3 → 4 → 5 → E. The process continues until the starting node is reached (Figure 5.14). The shortest path is now revealed A → B → E → G → F.

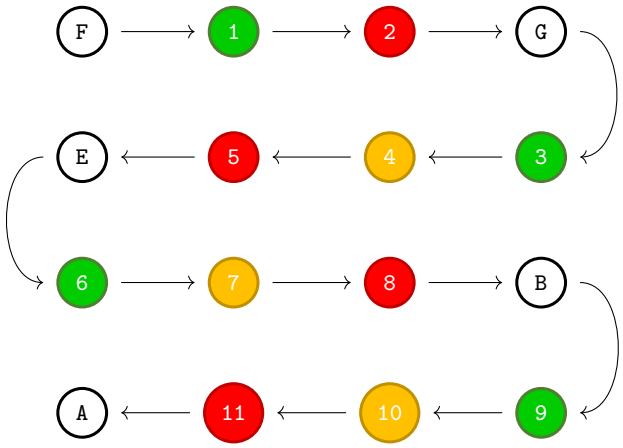


Figure 5.14 Snake diagram of shortest path deduced from computer generated weighted matrix table

A computer generated plot highlighting the calculated shortest (optimal) path between nodes A(1) and F(6) is shown in Figure 5.10b.

The results obtained from the preliminary analysis of the optimisation algorithm indicate that the computer code correctly identifies the shortest path between a source and target node. The next section, therefore, moves on to discuss the results when the same optimisation algorithm is applied in the context of decentralised energy management.

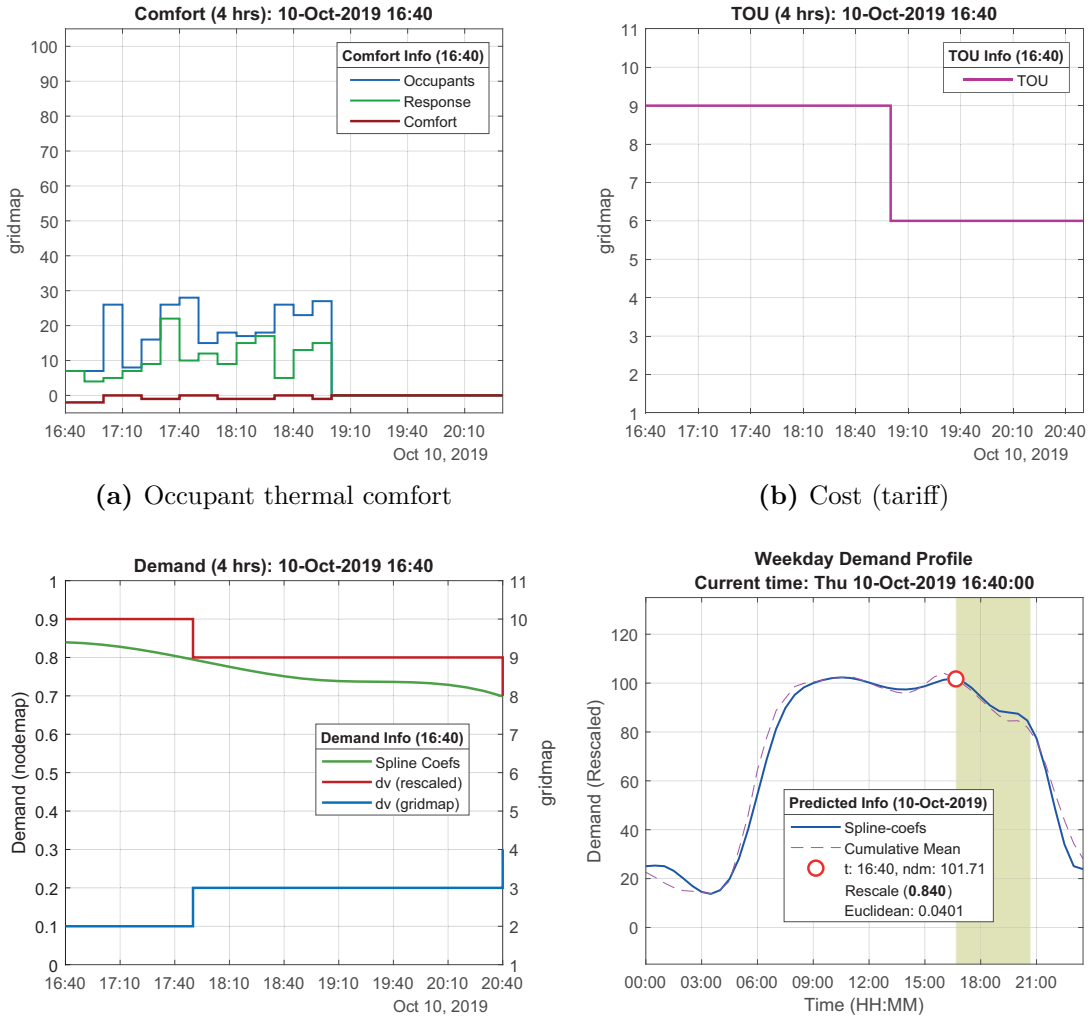
To begin, we evaluate the data input models. Individual charts created using nodemap data, and corresponding gridmap data validate the optimisation and control behaviour. In the second study, the results obtained from a simulated tertiary DR event are discussed. Finally, we monitor the system behaviour during an imbalance between supply and demand. Here, the pro-active frequency control reacts to a simulated load disturbance causing a frequency excursion from the nominal 50 Hz steady-state. The model is initialised using the values reported in Table 5.3.

Table 5.3
Computational model initialisation parameters

Parameter	Description	Value
`S0_date`	Stage 0 date time	`10-Oct-2019 16:00`
`des_begin`	Notification of DR event	`10-Oct-2019 16:40`
T^{th}_{min} (°C)	Minimum temperature threshold	16.6
T_{step} (°C)	Temperature step increase	3
T_{room} (°C)	Room temperature	18
Horizon (h)	Forecast horizon	4
DR_t (min)	Tertiary DR event duration	40
`SOC_hi`	SOC maximum threshold	0.8
`SOC_lo`	SOC minimum threshold	0.2

Occupant thermal comfort feedback is shown in Figure 5.15a. At 16:40 the model reports the aggregated occupant thermal comfort in a space is “too warm”. This consensus triggers the optimisation algorithm to set the comfort level gridmap trajectory on a path that reduces the measured room temperature by 0.5 °C, i.e., $S_n \xrightarrow{\eta} S_1$ where $\eta = T_{S_0} - 0.5$ °C. Also, according to local settings, the timetable sets the number of occupants in a space to zero at 19:00. A ‘no occupancy’ status has clearly defined adaptive triggers. Firstly, the comfort signal values (occupants, response, and comfort) are held at a constant zero, while the number of

occupants present in a space is zero. Secondly, at 19:00, the optimiser begins to alter the comfort level gridmap trajectory by reducing the temperature to a minimum temperature threshold T_{min}^{th} (local setting) at a rate of 0.5 °C every 10 min. This behaviour is confirmed in the optimiser gridmap visualisation and subsequent optimiser nodemap shown in Figure 5.16.



(a) Occupant thermal comfort

(b) Cost (tariff)

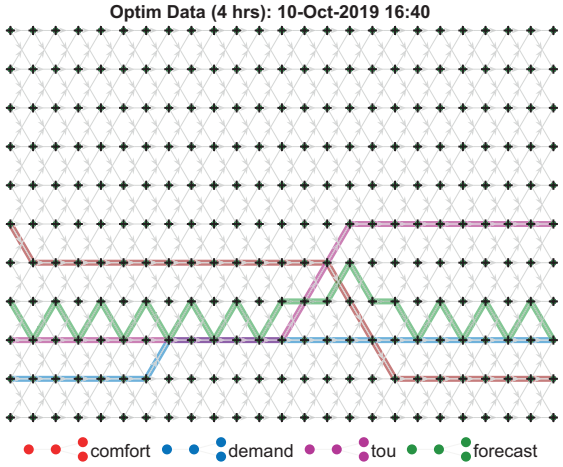
(c) Electricity demand

(d) Electricity demand forecast weekday 24 hr

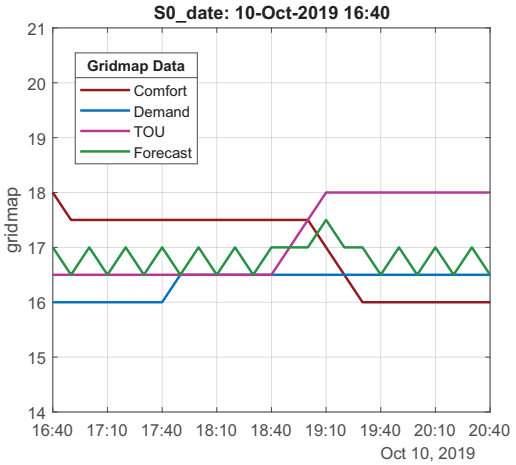
Figure 5.15 Gridmap visualisation of data type function response at 10-Oct-2019 16:40 over a 4 hr horizon window

The price in the three-tier TOU tariff is translated visually in Figure 5.15b. Initially, from 16:40 to 19:00 the TOU signal value is set to 9, which represents cost 24.99 p/kWh (peak), reducing to 6 (11.99 p/kWh mid-peak price) at 19:00. The energy cost nodemap data ($\delta_{\mathtt{ec}}$) transformation to the optimiser gridmap is shown in Figure 5.16. During peak periods, when the cost of energy is highest, the gridmap interpretation is to influence the control variable by reducing the temperature setpoint, which in turn reduces the cost of energy. Similarly,

at 19:00 (mid-peak), the gridmap tou signal is set at mid-scale (nominally 18 °C). The electricity demand forecast is shown in Figure 5.15d. To help interpret the demand signals shown, Figure 5.15c illustrates the calculated weekday demand profile over a 24 hr period. The red circle marks the start of the 4 hr horizon window (shaded area). The dv (gridmap) signal is reconstructed within the optimisation algorithm. The results are consistent with the modified layout of corresponding digraph object node coordinates, which describes the relationship between directional edges and connecting nodes shown in Figure 5.16a. The optimal temperature path is calculated at a sample rate of 10 min. Figure 5.16b highlights the optimal temperature value over a 4 hr horizon window commencing 16:40. The control action for the continuing 10 min cycle shown is the temperature value specified at 16:50, that is $T_{S_1} = 16.5\,°\text{C}$. This accords with our earlier occupant thermal comfort feedback report, which registered a consensus to reduce the room temperature by 0.5 °C.



(a) $31 \times 72 \times 4$ gridmap visualisation



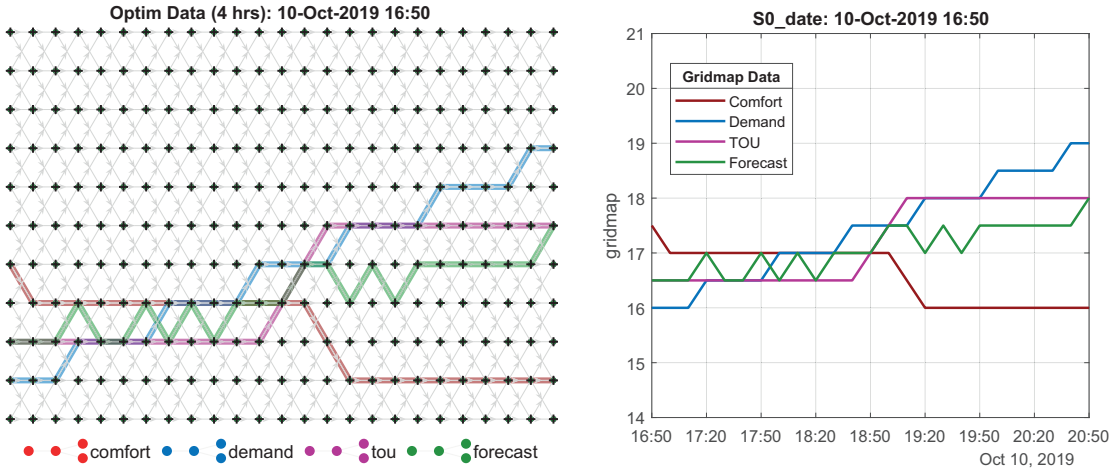
(b) Gridmap projected onto 11×25 nodemap

Figure 5.16 Optimisation response 10-Oct-2019 16:40

On receipt of a DR event notice (16:40) the normalised demand forecast value, $\delta_{\texttt{dv}}$ is recast to $\epsilon_{\texttt{dv}}$. The modified demand profile trajectory is defined by the Dijkstra's shortest path algorithm $\kappa_s \xrightarrow{\eta} \kappa_t$ where $\eta = T_{S_0} + T_{step}$ (°C). As can be observed from Figure 5.17, the change in demand profile at 16:50 increases from 16 °C (T_{S_0}) to 19 °C ($T_{S_{24}}$). A sample rate of 600 sec accounts for the slight delay from the start of the DR preparatory window to the change in demand profile trajectory. Although the supposed outcome is to promote an increase in temperature equivalent to T_{step} (°C) leading up to the start of the DR event, the projected valued is offset by the continued influence of the thermal comfort ($\epsilon_{\texttt{tc}}$) and energy

cost ($\epsilon_{\texttt{ec}}$) (tou) decision variables. Consequently, in this instance, the optimisation algorithm set the 4 hr ahead optimal temperature value slightly less than the anticipated 19 °C.

The layout of individual digraph objects and their corresponding nodemap representation, shown in Figure 5.16 and Figure 5.17 respectively, serve to provide a snapshot of the optimiser outputs overs a 4 hr horizon window any given time. The benefit of the optimiser is now translated into Figure 5.18, which plots several decision variables and control actions over a 24 hr period. Between Figure 5.18a and Figure 5.18b, we observe the impact of demand and tariff data on the temperature setpoint (`TS1`). Furthermore, the outside temperature (`Tout`) as no impact on the measured room temperature during this simulation.

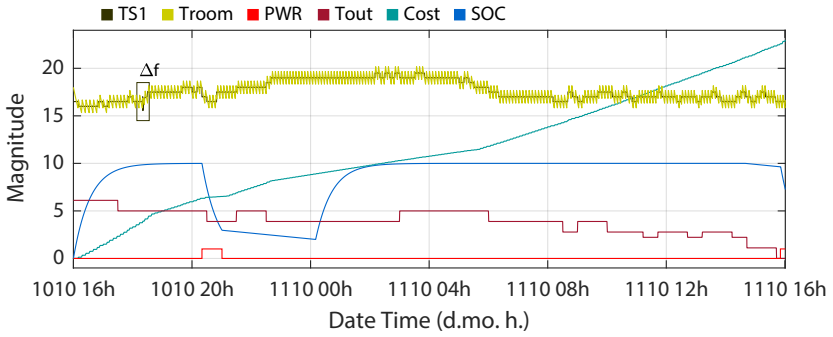


(a) $31 \times 72 \times 4$ gridmap visualisation

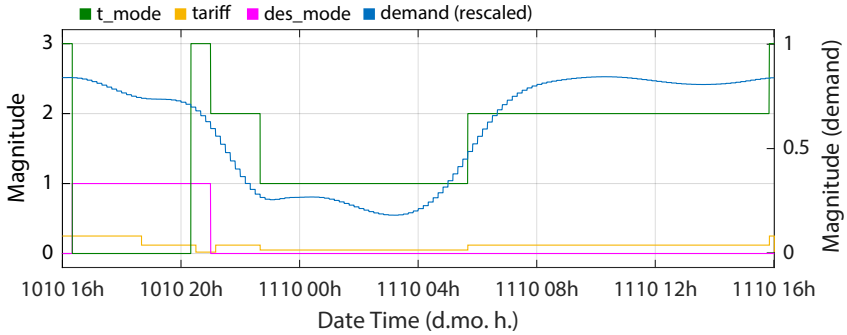
(b) Gridmap projected onto 11×25 nodemap

Figure 5.17 Optimisation response 10-Oct-2019 16:50

The start of the DR preparatory window is recorded at 16:40 and subsequently sets and holds `des_mode` = 1 for 4 hr and 40 min (the time leading up to and including the DR event). The BESS is seen to start a charge period in readiness to the start of the DR event. A tariff mode signal (`t_mode`) automatically restricts the use of the BESS until the DR event starts. At 20:40, the power signal (`PWR`) switches the primary power source from the grid to BESS. If the cost of energy is peak tariff immediately after the DR event (`t_mode` = 3), then the BESS would continue as the primary power source. However, as can be observed the BESS SOC signal (`SOC`) indicates the BESS starts a discharge phase at from the start of the DR event and continues, in this scenario, to the end of the DR event. At 21:20, the primary power source reverts to the grid, but the BESS remains available ($SOC > SOC_{lo}^{th}$).



(a) Temperature setpoint (°C) (`TS1`), room temperature (°C) (`Troom`), primary power switch signal (`PWR`), outdoor temperature (°C) (`Tout`), cost (p/kWh) (`Cost`) and BESS SOC (rescaled) (%) (`SOC`) profiles



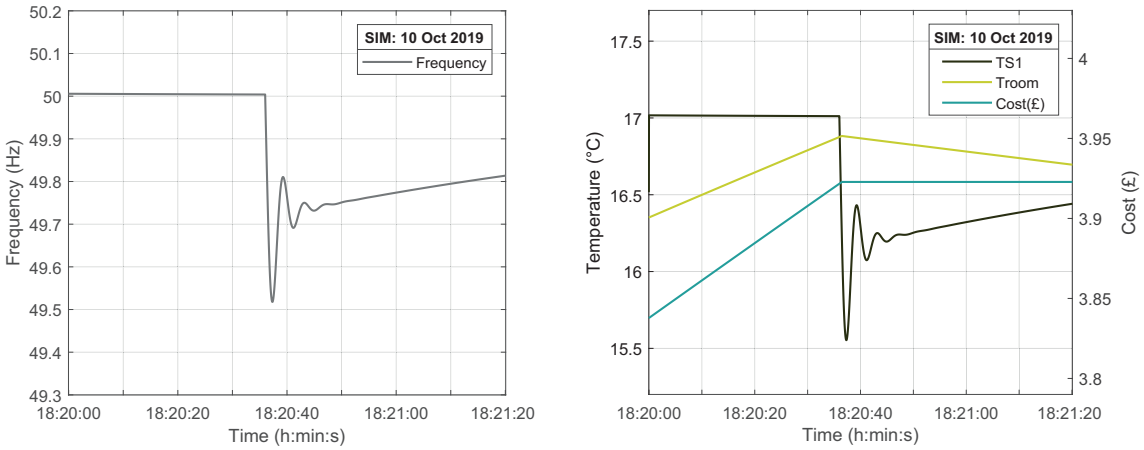
(b) Tariff mode (`t_mode`), TOU tariff (`tariff`), demand event signal mode (`des_mode`) and demand (rescaled) (`demand`) profiles

Figure 5.18 A simulation study at 10-Oct-2019 16:00 for 24 hr with DR event

The rate at which the energy source naturally discharges has been magnified to evaluate control actions when SDR exceeds low and high charge threshold values (local settings). In practice, SDR parameters should be set accordingly. The simulation results show the calculated electricity demand forecast profile (`demand`). Its impact on the optimisation algorithm is clear, when demand is high (06:00 to 22:00) the aggregated effect is to limit the temperature setpoint (reducing the demand for electricity on the distribution network). Conversely, when demand is low (22:00 to 06:00), the constraints that govern the temperature setpoint are relaxed. Here the optimiser allows, not mandates, an increase in energy consumption by increasing the space heating temperature setpoint. This finding, while preliminary, suggests the proposed control strategy has the potential to deliberately lessen peaks in demand (electrical) and fill in the period of low demand.

At 18:20.36, the impact of a simulated load disturbance ΔPd Table D.1 within the power subsystem is highlighted. The large and rapid decreasing frequency excursion shown in

the box highlight, signifying an imbalance between supply and demand, is observed more clearly in Figure 5.19a. The proposed system immediate response is to lower the temperature setpoint (T_{S_1}), reducing the on-site heat source energy consumption and thus providing a pro-active response to the stability of the electrical distribution network [5]. As can be observed in Figure 5.19b, and in the broader context in Figure 5.18a, these immediate interventions have minimal impact on measured room temperature (T_{room}), hence minimising occupant thermal discomfort.



(a) Impact on grid frequency due to simulated load disturbance

(b) Proposed model response

Figure 5.19 Frequency response

The accumulative effect of primary frequency regulation and system response to tertiary DR events can be observed in Figure 5.20. The design of the simulation model allows a comparison of baseline and simulated optimal behaviour. Here, energy costs follow a similar trajectory up to the start of a DR event at 20:20. A subsequent reduction in room temperature setpoint, while maintaining occupant thermal comfort, implies a decrease in energy consumption during the DR. Furthermore, this demonstrates that energy consumption has been shifted from the DR window.

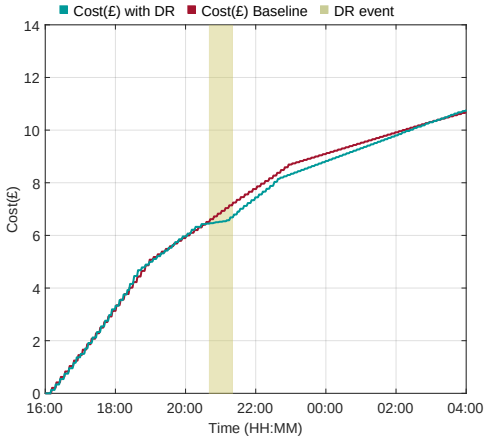


Figure 5.20 Electricity cost during demand event

5.11 Summary

The action of feedback systems cannot be described in terms of the aggregated behaviour of its forward path alone. According to the eight laws of software evolution, feedback constrains the behaviour of interconnected components and will modify their individual, local and collective performance [217]. If the software process fails to take into account change initiated in the surrounding environment when attempting to predict future outcomes, it is highly likely that the system will not perform in a manner that is consistent with the design or expectation. The simulation model technical development approach has observed this important principle. Reasonable decisions have been taken throughout the process and have been implemented accordingly. Feedback loops have attempted to deliberately address the performance gap that exists between building performance and model predicted behaviour. The shortfalls mentioned above have been considered when developing a new model, which aims to replicate the stochastic behaviour of building occupants. This contribution to the optimisation algorithm helps formulate a decision-making process that combines measured room temperature with other domain data. Collectively they represent the three significant data inputs to the optimisation algorithm, which is formulated using a weight-based routing algorithm. The contribution to research set out in this chapter is the development of an optimisation algorithm modified to support demand response services using three significant data inputs. A series of tests demonstrates the behaviour of a heating system has been altered by changing the temperature setpoint over a short horizon window based on the projected

energy demand, cost (tariff) and thermal comfort. The supporting control mechanisms put in place involves activities that aim to demonstrate active/pro-active response to changes in measured grid frequency and tertiary DR services.

Chapter 6

Case Study:

Optimisation and Control

6.1 Introduction

In this chapter, we apply the proposed optimisation and control algorithm described previously. We perform early deployment activities using prototype hardware in an experiment designed to evaluate the interaction of energy assets for optimal control with real-world data. Special attention is given to the control actions that underpin the usefulness of the proposed optimiser. This study aims to follow the test approach shown in Figure 6.1. The test cycle must observe the data and maintain the hardware and algorithm synchrony. The closed-loop testing environment we describe allows transition points between software-in-the-loop and hardware-in-the-loop activities. These act as the interface and provide a convenient and necessary breakpoint to complete any rework required. This chapter will begin with a description of designated hardware-in-the-loop simulations, including a review and selection of simulation software tools. Details of a smartphone app designed to allow building occupants to report relative thermal comfort levels are then presented before configuration and set up of experimental environment is described.

6.2 Experimental test environment design

The prime objective for the development of hardware-in-the-loop simulation is to advance a form of rapid prototyping that enables a detailed examination of the design problem (requirements) and move the design towards a satisfactory implementation. Early testing of the optimisation algorithm demonstrated the performance was adequate. Given the

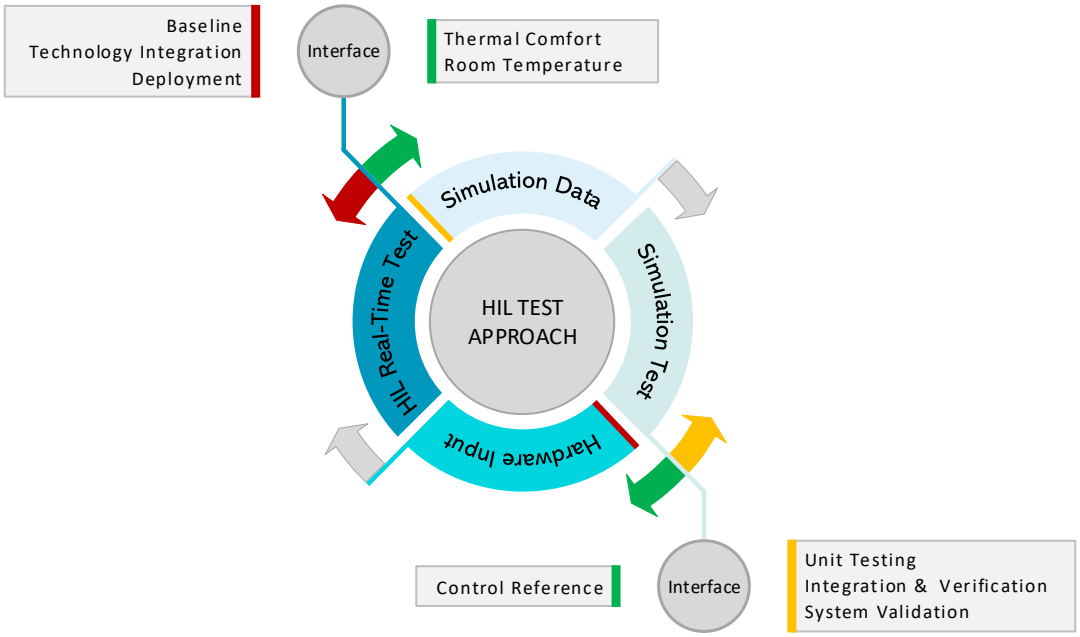


Figure 6.1 Hardware-in-the-loop test approach

favourable test performance of previous computer simulations, the experimental evaluation is seen as a legitimate progression in the overall test approach. During this phase, hardware, software, network requirements and test data requirements required to support the test environment are identified. Performance measure requirements and procedures to control the test configuration and environment are all considered. The primary objectives of performing tests are: (1) find defects, which may have been created during the development of software code, (2) gain confidence in the product, and (3) ensure the product satisfies declared test objectives.

Given one of the significant objectives of testing is to assess the integration of the optimiser software code by connecting other software and hardware components, a test environment was designed with the following main requirements in mind:

- The optimiser software module design requires the following data: measured room temperature, calculated demand (electrical) forecast, tariff (cost), notification of tertiary demand event and occupant thermal comfort feedback.
- The evaluation test is to be run in real-time.

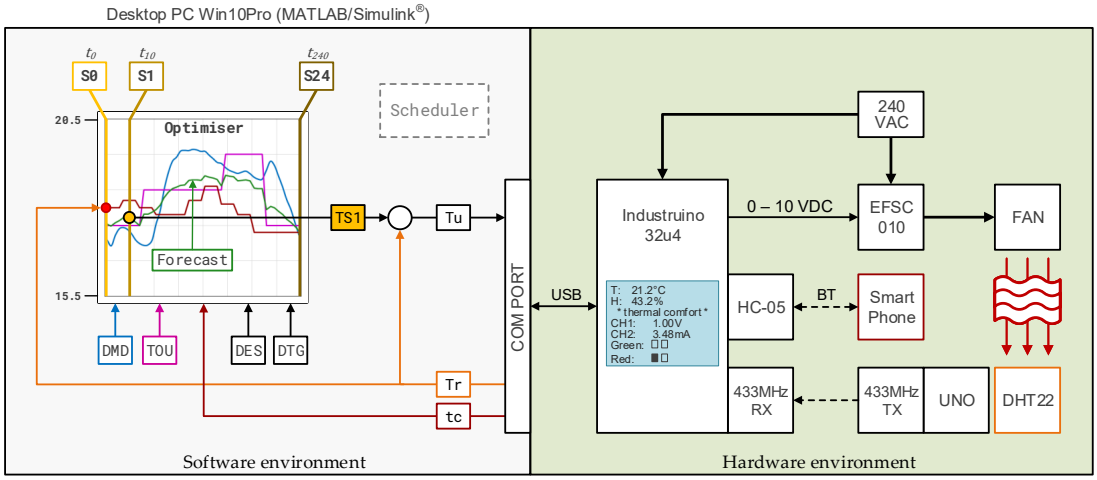


Figure 6.2 Abstract of optimisation algorithm and schematic diagram of the proposed hardware-in-the-loop test environment

- A suitable industrial type controller will provide an interface between the optimisation algorithm host computer and hardware components (e.g., heater, smartphone and temperature sensor).
- Testing will assess the optimiser response to a tertiary DR event, i.e., from the start of the DR preparatory window up to and including the demand event.
- Occupants must be able to send a message that describes their relative thermal comfort.
- Functionality that demonstrates switching of primary power source from grid-connected mode to a battery energy source will be simulated.
- Sampling time to update the optimiser control action signal is 10 min.

6.3 Simulation software and hardware selection

In this study, there is a clear separation between software and hardware environments (Figure 6.2). MATLAB/Simulink® was used to write software code for the optimisation algorithm, providing an interface between the simulation and hardware environment, emulating tertiary DR, and supporting in-test and post-test data analysis and visualisation. In addition, the open-source Arduino integrated development environment (IDE) was selected to write code and upload it to the industrial controller and supplementary Arduino products used during hardware-in-the-loop tests.

The test environment was designed to include the following major hardware items:

- Desktop PC Intel® Core™i5-3470 CPU @ 3.20 GHz 8.00 GB Win10Pro.
- Industruino IND.I/O controller (Arduino Leonardo ATmega32u4 microcontroller).
- MDR-20-24 power supply 24 VDC at 1 A.
- Climate King box fan heater 3 kW (HCK-BX3UK).
- CADAMP electronic fan speed controller (EFSC-010).
- Arduino Mega 2560 Rev3 (ATmega2560 microcontroller).
- Arduino Uno Rev3 (ATmega328P microcontroller).
- Arduino components for temperature sensing and data transfer from a remote sensor to a microcontroller.
- Android smartphone with Bluetooth capability to host thermal comfort feedback app.

6.3.1 Industruino IND.I/O D21G controller

The product hosts a ATmega32u4 microcontroller with 32 kB of flash. The 32u4 top board supports two analog output channels (4 to 20 mA/0 to 10 VDC). Industruino is Arduino compatible, housed in a DIN-rail mountable case and includes an on-board LCD with membrane switch panel (Figure 6.3). The baseboard provides a viable solution to bridge the gap between Arduino and industrial type sensors and actuators. In this study, the microcontroller is configured to operate as a bridge between the host WinPC and hardware equipment. The 0 to 10 VDC output regulates the heater airflow and the second channel (4 to 20 mA) is configured (for demonstration purposes only) to provide a visual indication of measured room temperature. A series of digital pins (CH2, CH3, CH4 and CH5) provide visual feedback of smartphone status. The microcontroller unit (MCU) connect to USB UART, 433 MHz RXD (remote temperature sensor) and HC05 (smartphone Bluetooth) equipment. The Industruino IND.I/O is powered by a separate 20 W 24 VDC single output industrial DIN-rail power supply.

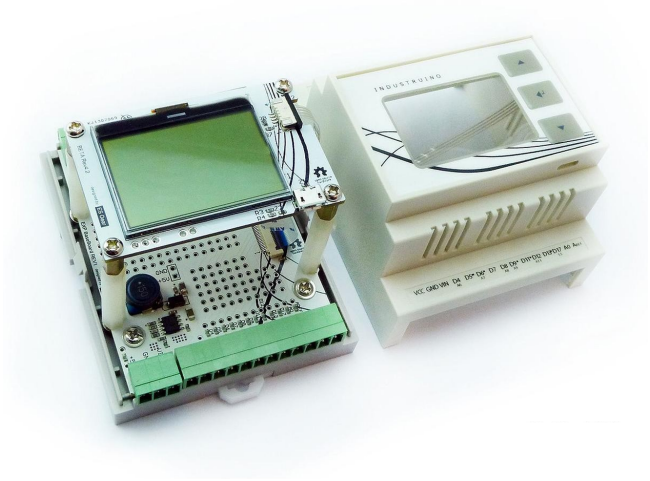


Figure 6.3 Industruino IND.I/O D21G

6.3.2 MDR-20-24 power supply

This product features a universal AC input (85 to 264 VAC) and 24 W 24 VDC output. Protections include short circuit, overload and overvoltage. In this study, the MDR-20-24 24 VDC power supply is used to power the Industruino IND.I/O D21G controller.

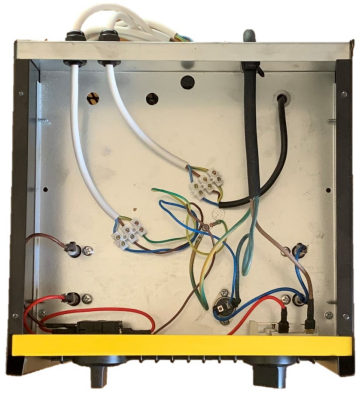
Figure 6.4 Mean Well MDR-20-24 power supply

6.3.3 Climate King box fan heater

Climate King box fan heater 3 kW (HCK-BX3UK) (Figure 6.5a). The internal equipment includes a single-phase shaded pole induction motor (Part No. YZF482175A 25 W 1300 rpm 220 to 240 Vac) which is used to drive a fan to regulate the airflow. The unit allows the fan to cool the equipment after the heating element has been turned off. A thermostatic cut-off feature will cut the power if the heater gets too hot for too long. In this scenario, to regulate the airflow, the factory-installed wiring for the heater control mechanism has been

modified to allow the external control signal to adjust the shaded pole induction motor, see Figure 6.5b.

(a) Heater unit

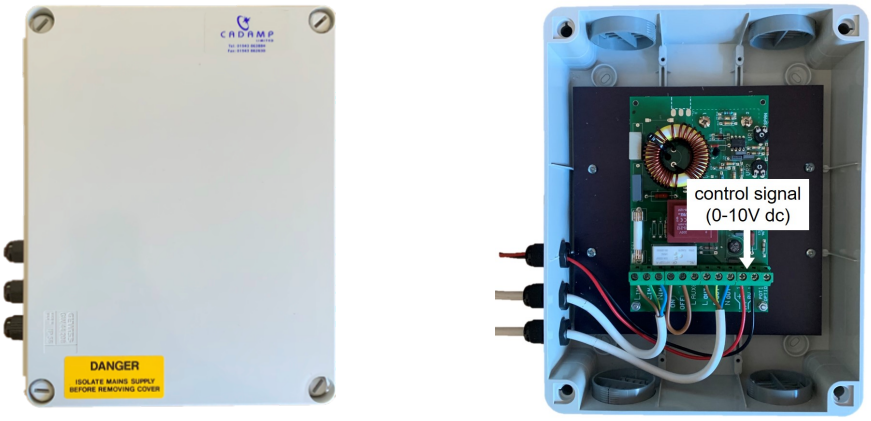


(b) Modified internal wiring layout

Figure 6.5 Climate King 3 kW box fan heater

6.3.4 CADAMP electronic fan speed controller

CADAMP electronic fan speed controller (EFSC-010). Designed to accept a 0 to 10 VDC input signal and can control the speed of the heater single-phase shaded pole induction motor accordingly (Figure 6.6a). Internal wiring is configured to operate using a remote speed adjustment station (fitting a link between terminals 4 and 5). A 0 to 10 VDC input control signal originating from the optimisation algorithm is connected to terminals 10 (`+I/P`) and terminal 11 (`0VI/P`). Electrical supply 230 V 1PH 50 Hz input to terminals 1, 2 and 3 respectively and finally terminals 10, 11 and 12 connect to the remote speed adjustment station (thermostatically controlled load), see Figure 6.6b.



(a) Modified case unit **(b)** Wiring control circuit

Figure 6.6 CADAMP electronic fan speed controller

6.4 Arduino software development

The Arduino family of components utilised in this research provide three primary services. Software code has been developed for each service using the Arduino IDE. Table 6.1 details each service, its corresponding sketch name and host Arduino board. Code listings are documented in Appendix C.

Table 6.1
Arduino software services

Service	Description	Sketch Name (Board)
1	Interface between MATLAB/Simulink® software environment (Desktop PC Win10Pro) and hardware environment	`control_unit.ino` (Leonardo)
2	Send and receive messages from one Arduino to another Arduino board using 433 MHz	`transmitter.ino` (Uno Rev3)
3	Provide test stub that simulates the behaviours of measured grid mains frequency[1]	`mains_frequency.ino` (Mega 2560 Rev 3)

A schematic diagram of the Arduino sketch `control_unit.ino` development complete with external input and outputs and internal outputs is shown in Figure 6.7.

[1] Arduino software service No.3 is detailed as an optional alternative. The Arduino Mega 2560 R3 (complete with installed sketch `mains_frequency.ino`) is designed to output a continuous loop of grid measured frequency (255 data recorded from BMRS data at sample time equal to 1 sec). It emulates the frequency measurement instrument described at Appendix A.

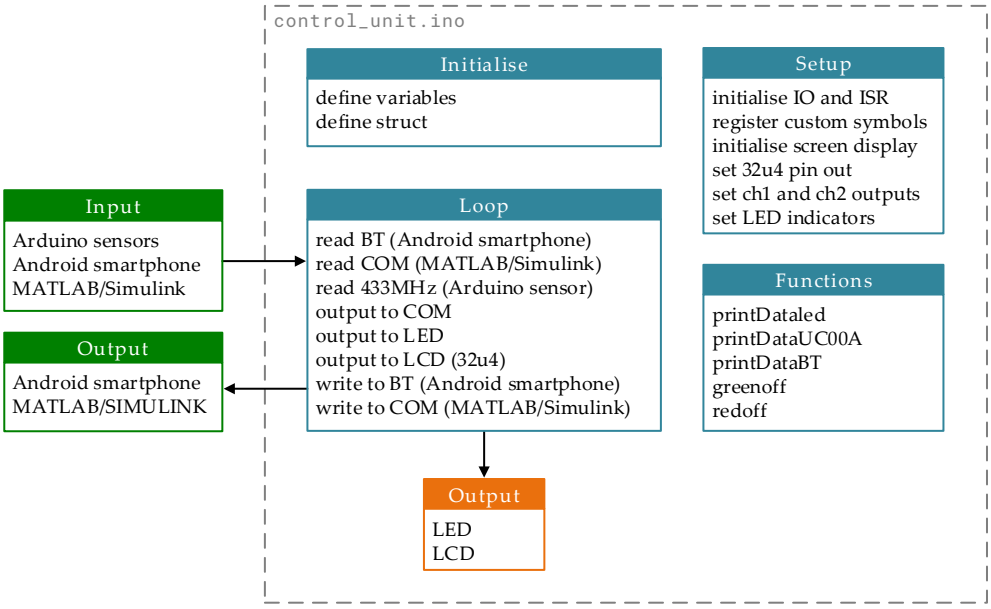


Figure 6.7 Schematic diagram of Arduino sketch development - control_unit.ino

6.5 Building occupant engagement

Improvements in building energy efficiencies can be attributed to technological advances in architectural design, material, and technology. However, there is quite often a disparity between the predicted energy performance of buildings and actual energy usage [218]. According to recent studies, the scale of this so-called performance gap means actual energy consumption can be up to five times the predicted (computed) value [219]. This discrepancy is not exclusively attributed to the evaluation of building performance where results are based on improved physical transformations and formulation of energy-saving standards. Instead, the main reason is that the prediction models crude (or sophisticated) simplification of the physical environment. Among these uncertainties, building occupant behaviour has been identified as one of the significant contributing factors when developing a prediction model for measuring building performance [220, 221]. Today, thermal comfort is defined as 'that condition of mind that expresses satisfaction with the thermal environment' in the globally recognised ASHRAE 55 (see, [222]) and ISO 7730 (see, [223]) standards for evaluating indoor environments. These comfort limits can be expressed by the predicted mean value (PMV) or predicted percentage of dissatisfaction (PPD) indices. Improved modelling techniques can help align model behaviour with the physical world. In terms of human behaviour, this can

translate to knowledge about building occupancy schedules, which provide time-dependent occupancy details [224]. While this information will help calibrate the prediction model, additional information is required. Participatory control through occupant feedback has been a more recent development in predicting building performance. In the optimisation model offered in Chapter 5, an algorithm was developed to predict an average thermal comfort sensation of a group of people. This method included knowledge of building occupancy during a set number of time intervals of each weekday. While this plays an essential role in improving the optimiser, the most critical development was establishing feedback from occupants in near real-time.

In recent studies, it is reported that the most common type of interface used for a collection of thermal comfort feedback was based on basic mobile applications [221]. Utilising these types of technologies is more compatible with real-life scenarios than methods that make assumptions of thermal comfort, which are based on more generalised estimates of occupant preferences [225]. The so-called personal comfort models, where thermal comfort is recognised as being subjective and a matter of personal preference, are more likely to reflect individual thermal comfort preferences than generalised group assumptions [226]. The approach used in this study acknowledges technology-mediated thermal comfort feedback is a valuable energy management intervention. Different thermal demands and occupants diversified preferences may lead to low occupant satisfaction rates. However, despite a plethora of research on this subject, it is difficult to ascertain which factors should be included in personal comfort models [227].

In this study we aim to remain sensitive to the complex subject of indoor thermal comfort. However, we are more interested in the contribution of occupant thermal comfort as a condition of state and its subsequent impact on energy management. More specifically, using averaged individual thermal comfort reports to influence the optimal path in energy management. In doing so, the proposed optimisation algorithm has been configured to react to a model that considers a consensus of multiple individuals when examining indoor thermal comfort requirements.

In this experimental phase of work, we propose a general conceptual model that emulates a two-way communication process through Bluetooth to allow processing of information and

collection of user thermal comfort feedback. This approach lets us focus on the process-nature of communication with building occupants. Also, it enables us to validate all the input and outputs branches of each stage, including the related instruments that the technology offers. The expectation is thermal feedback reports can be submitted at any time. However, a more compatible strategy is to align data collection on a just in time basis, i.e., at the beginning of each optimisation cycle at a sample rate of 10 min.

Identifying the best technology from a set of possible alternatives is a technology selection problem. Several factors need to be considered during any selection process [228]. Advanced technologies mean selecting the right techniques can be even more challenging. Nevertheless, since the objective of this experiment is to demonstrate occupant engagement, a smartphone app was considered a suitable platform to allow occupants to record the condition of individual thermal preferences.

Table 6.2 lists several basic functional (F) and non-functional (NF) requirements. The list helps focus on the development progression and build a basic testable and traceable requirement specification. The smartphone app demonstrator application is strictly limited to support the experimental study. Therefore maintainability, security, cultural, political, and legal requirements are not specified.

Table 6.2
Smartphone app basic requirements

Item	Type	Description
1	F	The product shall record an individual's thermal comfort preference
2	F	The product shall send thermal comfort preference to energy management system
3	F	The app shall provide a suitable user interface that allows the individual to record thermal comfort preference
4	NF	The app user interface shall provide a visual indication of selected thermal preference, location, and actual room temperature
5	NF	The product will allow individuals to exchange data between fixed energy management system and mobile device (smartphone) using wireless technology
6	F	The energy management system shall process all received thermal comfort reports based on the most recent data made available

6.5.1 Application development

The entire smartphone app was designed to operate on an Android smartphone through the MIT App Inventor 2, which uses block-based programming language built on Google Blockly [229] and inspired by languages such as StarLogo TNG [230] and Scratch [231]. Figure 6.8 shows a schematic of the smartphone app model with inputs and outputs.

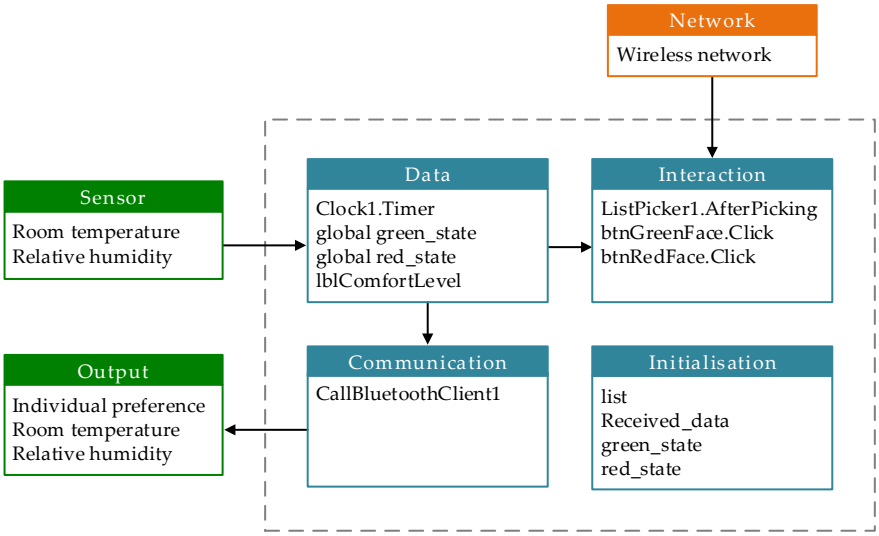
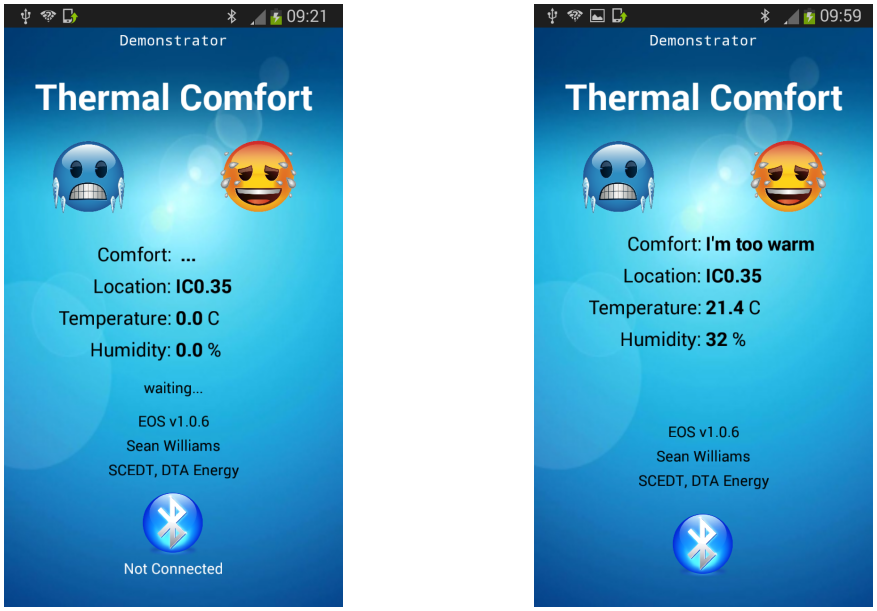


Figure 6.8 Schematic diagram of app model with inputs and outputs

A prototype application user interface design included two 3-stage buttons, which allow the operator to cycle through different thermal comfort preferences: (1) I'm too cold, (2) I'm cold, (3) I'm okay, (4) I'm hot, and (5) I'm too hot. A text banner message reports the selected condition status. A further button (Bluetooth icon) allows the individual to connect to the wireless network. An appropriate text banner message confirms the status of the connection. The app start screen is shown in Figure 6.9a. After network connection has been achieved, the individual can begin to register their thermal comfort preference. Figure 6.9b.



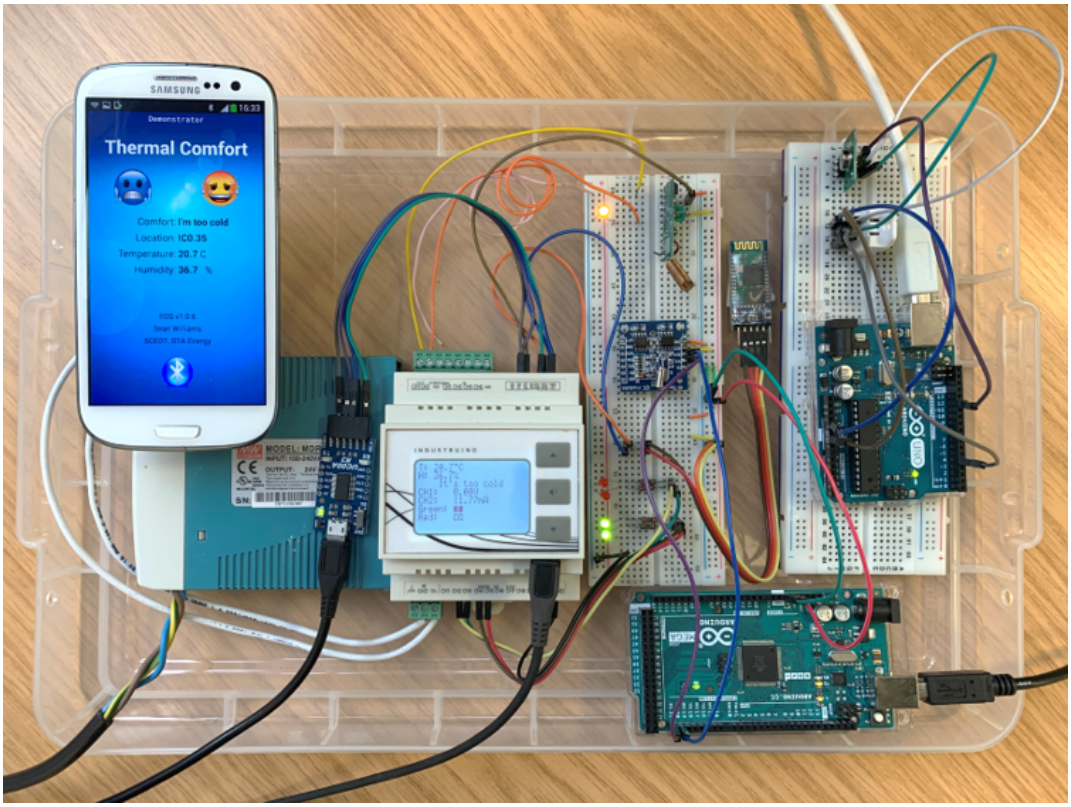
(a) App not connected **(b)** App connected

Figure 6.9 Smartphone app screen images

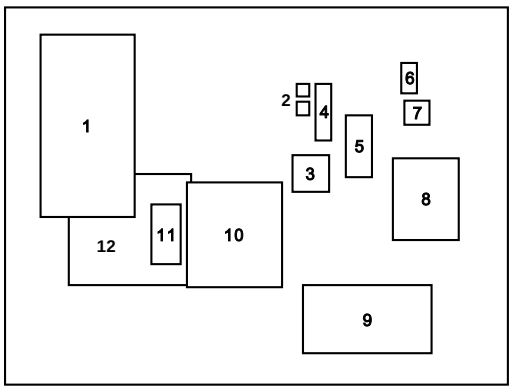
The program logic (Figure 6.8) is divided into four groups: (1) Initialisation, (2) Interaction, (3) Data and (4) Communication. The corresponding app block code is listed in Appendix C.

6.6 Experimental test design and set up

A complete wiring diagram of the hardware-in-the-loop test environment is shown in Appendix C. For the indoor thermal environment measurement, a wireless temperature sensor is deployed to a suitable fixed position in the room. An image of Arduino components and smartphone before the positioning of sensors is shown in Figure 6.10. Equipment schematic and legend are also provided. The diagram excludes major equipment items listed earlier, i.e., host computer, heater, and electronic fan speed controller.



(a) Equipment connection (benchtop)



(b) Equipment schematic

Item	Description
1	Android Smart Phone: Bluetooth TXD
2	Analog Output 2x CH (4-20 mA/0-10 VDC)
3	Real Time Clock: RTC DS1307 i2C Module
4	433 MHz RF: Wireless Data Receiver
5	HC05 Serial Pass-through Module: Bluetooth RXD
6	433 MHz RF: Wireless Data Transmitter
7	DHT22: Temperature and Humidity Sensor
8	Arduino UNO Rev3
9	Arduino Mega2560: Mains Frequency Test Harness
10	Controller: Industruino IND.I/O D21G
11	USB to UART: Data Transfer to MVS
12	MDR-20-24 Power Supply 24 VDC at 1 A

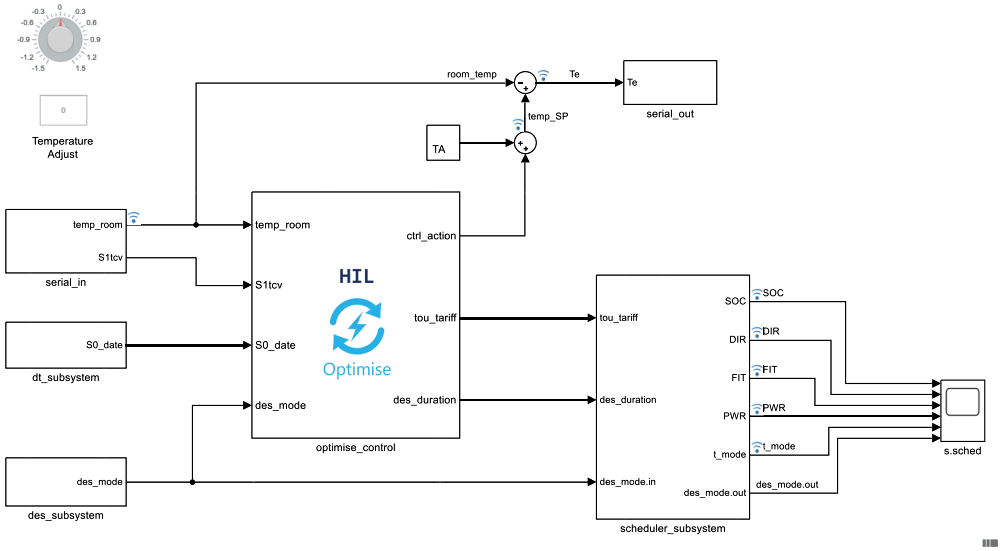
(c) Equipment legend

Figure 6.10 Hardware-in-the-loop test environment

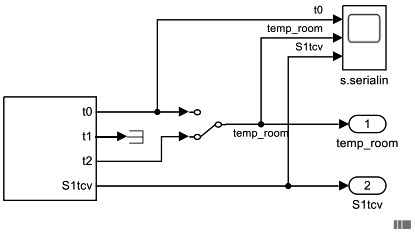
6.7 Simulation model update

The desktop simulation model for energy optimisation framework developed in Chapter 5 is modified to provide a bridge between the software and hardware environments. The revised

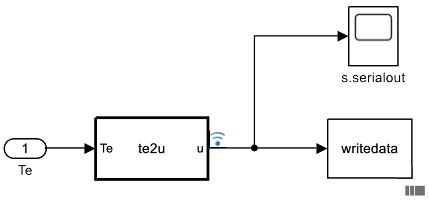
model shown in Figure 6.11a excludes the building and energy subsystems and introduces two new subsystems: serial_in (Figure 6.11b) and serial_out (Figure 6.11c).



(a) Energy optimisation framework



(b) Serial in subsystem



(c) Serial out subsystem

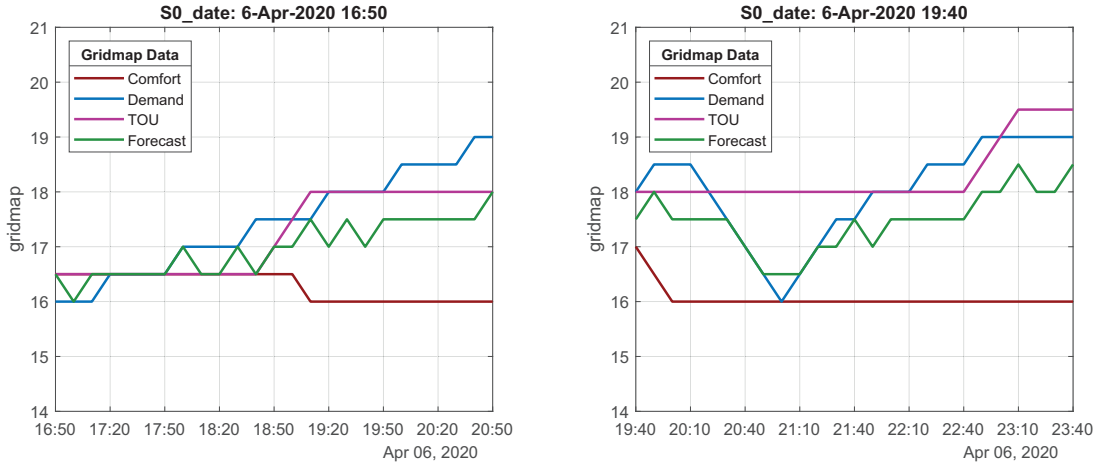
Figure 6.11 Computer model modifications for energy management

In addition to the changes made to the Simulink® model block layout, several software code updates and new functions are required. Table provided at Appendix C.6 summarises relevant code changes.

6.8 Experimental evaluation

Figure 6.12 shows the results of a preliminary test that was carried out in real-time. The test started on Monday 6th April 2020 16:00. At 16:40 the start of a DR preparatory event triggers a preset sequence of control actions designed to prepare the heating services

in advance to the 40 min DR event, which started at 20:40. The test was run for 5.5 hr, finishing 10 min after the DR event. Comparison of the findings shown in Figure 6.12 with those of earlier computation studies confirms the operation of the optimisation algorithm is consistent with our mathematical arguments, which posits that the interaction between declared data types can influence an environment space heating. Increasing the temperature setpoint successively by 0.5 °C at 10 min intervals during the DR preparatory stage increased the space temperature by 2 °C from the start of the DR preparatory window. Figure 6.13a confirms a temperature value of 18.5 °C was recorded at approximately 19:10. It can be observed the temperature then decreased to 17.4 °C at 20:40, which is the start time of DR event. This behaviour may be explained by the fact that the thermal comfort profile (`Comfort`) reduced to an equivalent of 16 °C (T^{th}_{min}) at 19:00; which is consistent with an expected zero occupancy at the same time.



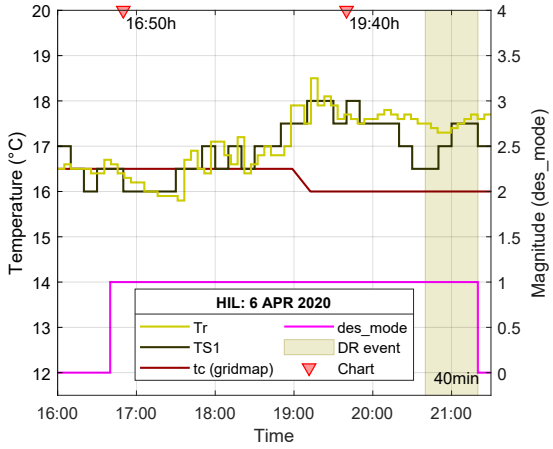
(a) On receipt of a simulated DR event signal at 16:50

(b) During the 40 min DR event (20:40 to 21:20)

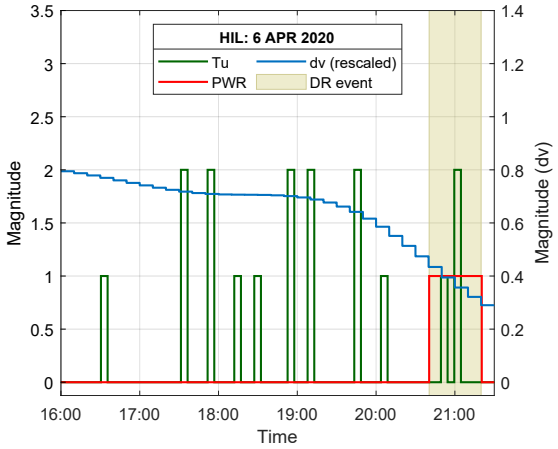
Figure 6.12 Visual representations of gridmap data showing 4 hr horizon window of predicted values of each data type and optimised temperature profile

Furthermore, as can be observed in Figure 6.13b, the control action signal utilised in the earlier computational study has been modified to regulate the physical heat transfer through flow. Here, the control action signal (`Tu`), which operates a 0 to 10 VDC EFSC, is proportional to the difference between the calculated optimal temperature setpoint (`TS1`) and the measured temperature (`Tr`), i.e., $Tu \propto Te$, where $Te = T_{S_1} - T_{room}$. The power switch signal (`PWR`) shows the virtual energy storage system is activated at 20:40 and continues to operate as the heating system primary energy source for the duration of the DR event

(shaded area). Overall, these results are very encouraging. The experimental evaluation raises the possibility that the proposed optimisation algorithm may support small island communities in a decentralised environment with limited access to communication networks.



(a) Room temperature (`Tr`), temperature setpoint (`TS1`), thermal comfort gridmap data (`tc`), demand event signal mode (`des_mode`) and DR event



(b) Control action signal (`Tu`), primary power switch signal (`PWR`), demand (rescaled) (`dv`) and DR event

Figure 6.13 Experimental evaluation recorded results at 6-Apr-2020 16:00 for 5.5 hr with DR event

6.9 Summary

The test environment offers a low-cost platform to validate the optimisation algorithm in real-time. For occupant thermal comfort preferences, a prototype smartphone app has been developed and installed on an Android operating system, which satisfies the requirements listed in Table 6.2. The computer simulation model has been modified to bridge the gap between the software and hardware environments. Industrial standard equipment provides an interface to electronic fan speed controllers, which in turn regulates heater airflow. Feedback signals inform the optimisation algorithm about room temperature and thermal comfort preferences.

New developments are required when attempting to explore the use of the proposed optimisation and control when deploying in-the-field energy management solutions. However, the functionality and proposed hardware-in-the-loop test environment provide a useful and meaningful step that aims to validate the technical framework offered. A structured approach

to equipment selection and testing means that the groundwork is in place to progress to more challenging user-engagement and large-scale replication.

Chapter 7

Conclusions and Recommendations

7.1 Introduction

This thesis advances knowledge for demand response services in community energy management. Changing room temperature in buildings is achieved by framing an optimisation problem that requires grid frequency measurement, energy forecasting, and knowledge of both spatial and temporal constraints. This chapter begins by reflecting on aims and methods first introduced in Chapter 1, summarising main findings and how each significant work activity has contributed to research objectives. Based on these findings, recommendations for future research brings this thesis to a conclusion.

7.2 Demand response in buildings

A decentralised frequency control regulation method has been validated in Chapter 3 using a series of computer models, a frequency measurement instrument, and a real controllable thermal load. Here, a demand response approach established regulatory control of room temperature through mechanisms that automatically respond to measured grid frequency and in response to explicit tertiary DR event signals. Hypothesis tests provided evidence that substantiates claims that data collected using the prototype frequency measurement instrument is as good as data from the National Grid.

Several interesting conclusions can be drawn from the results presented. They suggest that small excursions in measured temperature from setpoint values will not compromise indoor comfort but can contribute to the restoration of frequency equilibrium during network stress events. These findings mean that the utility of a decentralised demand response strategy could close the gap in reserve capacity margins availability by exploiting coupling technologies with near-zero intervention from the consumer. Furthermore, the approach presented offers a viable alternative to more traditional system balancing services that tend to be reactive (on/off) at set threshold values.

Chapter 4 documented a new mathematical model that uses a series of simple data transformations to provide a useful representation of demand time series. Designed to operate independently without the need to maintain an estimation dataset means, the simplicity of this approach allows for rapid deployment of future modification to the polynomial coefficients, thus ensuring its longevity. This finding suggests that the behaviour of existing energy optimisation technologies may benefit from similar approaches.

7.3 Conclusions from an experimental study

Overall, the results presented are very encouraging. The experimental evaluation raises the possibility that the proposed optimisation algorithm may support small communities in a decentralised environment with limited access to communication networks. Comparison of the findings with other studies confirms the novelty of the proposed demand response

scheme for energy management. It is encouraging, elements of this research are consistent with results found in previous work. Eriksson et al. [232] developed a normalised weighted constrained multi-objective meta-heuristic optimisation algorithm to consider economic, technical, socio-political and environmental objectives. The results emphasised the application of a modified PSO algorithm to optimise a renewable energy system of any configuration.

The implementation of the Dijkstra's algorithm (used in this study) is more prevalent in other applications (e.g., see [233–235]). Nevertheless, its simplicity makes it a versatile heuristic algorithm. A shortest path optimisation algorithm was designed to compute an optimal water heating plan based on specific optimality criteria and inputs [236]. The significant feature reported of the proposed algorithm was its low computational complexity, which opens the possibility to deploy directly on low-cost embedded controllers.

In a further study, a strong relationship between optimisation and space heating has been reported [237]. Here, a neural network algorithm was used to build a predictive model for the optimisation of a HVAC is combined with a strength multi-objective PSO algorithm. Although results show satisfactory solutions at hourly time intervals for users with different preferences, demand response mechanisms have not been considered. However, leveraging upon the concepts of Industry 4.0, Short et al. [152] demonstrated the potential to dispatch HVAC units in the presence of tertiary DR programme, can deliver satisfactory performances.

Finally, a more comprehensive study proposed an optimisation model which takes total operational cost and energy efficiencies as objective functions [238]. Here, a thermal load is adjusted in the knowledge that a managed change in temperature value has no significant impact on user comfort. An integrated demand response mechanism is also considered. Although the results provide a new perspective for integrated energy management and demand side load management, there is no further exploitation in real-time user engagement or perspectives on decentralisation.

7.4 Key findings

The aim of this study is to advance an integrated demand response in the decentralised community energy (electrical) system. So far, most studies have focused on specific

optimisation problems, more recently using complex algorithms that require uninterrupted access to data and high computational resources to function. In this study, attention has shifted more towards optimisation and control of community energy management. The approach addressed some shortfalls in literature. In doing so, this study emphasises the following contributions:

1. That the new arrangement reveals something useful, by demonstrating the operation of a prototype low-cost, standalone grid frequency measurement instrument [215].
2. There is no information available that describes simultaneous active/pro-active control of thermostatically controlled loads in buildings. The demand response offered provides active control of room temperature using a multi-objective cost function formulated using a weight-based routing algorithm [239]. The frequency measurement instrument provides pro-active demand response [5].

7.5 Recommendations for future work

Energy system integration is considered a crucial element of the European Commission's initiative to achieve climate neutrality by 2050. Its recent 'energy system integration strategy' rests on three pillars: a more circular energy system, with energy efficiency at its core; electrification of heating and vehicles; and use of low carbon renewable where direct heating and electrification is not feasible [240]. Here, energy efficiency is highlighted as an essential contribution to achieving an integrated energy system.

The EU includes more than 550 inhabited islands [241]. Despite access to renewable energy technologies, many continue to rely on electricity been generated using fossil fuel. The intermittent behaviour of some RES continues to impact grid stability. This study does not proclaim to reduce fossil fuel dependency or promote energy self-reliance. However, it may offer more communities an opportunity to debate an energy transition from a fossil-fuelled based platform towards a more sustainable low-carbon power system.

Firstly, to deploy the demand response regime, at scale, as part of a wider community-led energy management scheme is recommended. This action requires training the proposed demand forecasting algorithms using data from potential new deployment sites.

Secondly, a decentralised demand response in community energy system must prioritise the integration of new technological innovations in line with local community needs and aspirations. Therefore, new work is required that promotes energy citizenship by encouraging greater participation in community decision-making at the same time as helping foster sustainable energy use. In this context, validating the design and implementation of consumer feedback scheme (e.g., smartphone app) that reports user's participation in any local demand response initiative.

References

[1] T. J. Foxon, "Transition pathways for a UK low carbon electricity future," *Energy Policy*, vol. 52, pp. 10–24, 2013.

[2] D. Lara-Arango, S. Arango-Aramburo, and E. R. Larsen, "Uncertainty and the long-term adequacy of supply: Simulations of capacity mechanisms in electricity markets," *Energy Strategy Reviews*, vol. 18, pp. 199–211, 12 2017. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S2211467X17300639

[3] A. Pepiciello and A. Vaccaro, "An optimization-based method for estimating critical inertia in smart grids," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 2019.

[4] J. Villar, R. Bessa, and M. Matos, "Flexibility products and markets: Literature review," *Electric Power Systems Research*, vol. 154, pp. 329–340, 2018.

[5] S. Williams, M. Short, and T. Crosbie, "On the use of thermal inertia in building stock to leverage decentralised demand side frequency regulation services," *Applied Thermal Engineering*, vol. 133, pp. 97–106, 2018. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1359431117370242

[6] R. E. Bellman, "The Theory of Dynamic Programming," 1954.

[7] N. Gunningham, "Managing the energy trilemma: The case of Indonesia," *Energy Policy*, vol. 54, pp. 184–193, 2013.

[8] Word Energy Council - WEC, "World Energy: Trilemma Index 2019," World Energy Council, Tech. Rep., 2019.

[9] B. Alcott, "Jevons' paradox," *Ecological Economics*, vol. 54, pp. 9–21, 2005.

[10] S. Copiello, "Building energy efficiency: A research branch made of paradoxes," *Renewable and Sustainable Energy Reviews*, vol. 69, pp. 1064–1076, 2017.

[11] T. M. Harris, J. P. Devkota, V. Khanna, P. L. Eranki, and A. E. Landis, "Logistic growth curve modeling of US energy production and consumption," *Renewable and Sustainable Energy Reviews*, vol. 96, pp. 46–57, 2018.

[12] J. Freire-González and I. Puig-Ventosa, "Energy efficiency policies and the Jevons paradox," *International Journal of Energy Economics and Policy*, vol. 5, no. 1, pp. 69–79, 2015.

[13] M. Vujanović, Q. Wang, M. Mohsen, N. Duić, and J. Yan, "Special issue of applied energy dedicated to SDEWES conferences 2018: Sustainable energy technologies and environmental impacts of energy systems," *Applied Energy*, vol. 256, no. 113919, 2019.

[14] M. A. Abdullah, A. P. Agalgaonkar, and K. M. Muttaqi, "Assessment of energy supply and continuity of service in distribution network with renewable distributed generation," *Applied Energy*, vol. 113, pp. 1015–1026, 2014.

[15] K. S. Ratnam, K. Palanisamy, and G. Yang, "Future low-inertia power systems: Requirements, issues, and solutions - A review," *Renewable and Sustainable Energy Reviews*, vol. 124, no. 109773, 2020.

[16] A. A. Bayod-Rújula, "Future development of the electricity systems with distributed generation," *Energy*, no. 34, pp. 377–383, 2009.

[17] U. D. Neelawela, E. A. Selvanathan, and L. D. Wagner, "Global measure of electricity security: A composite index approach," *Energy Economics*, vol. 81, pp. 433–453, 2019.

[18] S. Kosai and H. Unesaki, "Short-term vs long-term reliance: Development of a novel approach for diversity of fuels for electricity in energy security," *Applied Energy*, vol. 262, no. 114520, 2020.

[19] IEA Statistics, "The World Bank," 2020. [Online]. Available: https://data.worldbank.org/indicator/EG.USE.COMM.FO.ZS

[20] J. Price, M. Zeyringer, D. Konadu, Z. Sobral Mourão, A. Moore, and E. Sharp, "Low carbon electricity systems for Great Britain in 2050: An energy-land-water perspective," *Applied Energy*, vol. 228, pp. 928–941, 2018.

[21] A. Hope, T. Roberts, and I. Walker, "Consumer engagement in low-carbon home energy in the United Kingdom: Implications for future energy system decentralization," *Energy Research and Social Science*, vol. 44, pp. 362–370, 2018.

[22] J. Barrett, T. Cooper, G. P. Hammond, and N. Pidgeon, "Industrial energy, materials and products: UK decarbonisation challenges and opportunities," *Applied Thermal Engineering*, vol. 136, pp. 643–656, 2018.

[23] M. Pena-Cabrera, V. Lomas, and G. Lefranc, "Fourth industrial revolution and its impact on society," in *IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, CHILECON 2019*. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 1–6.

[24] L. Dogaru, "The Main Goals of the Fourth Industrial Revolution. Renewable Energy Perspectives," *Procedia Manufacturing*, vol. 46, pp. 397–401, 2020.

[25] J. Busch, T. J. Foxon, and P. G. Taylor, "Designing industrial strategy for a low carbon transformation," *Environmental Innovation and Societal Transitions*, vol. 29, pp. 114–125, 2018.

[26] N. O. Bonsu, "Towards a circular and low-carbon economy: Insights from the transitioning to electric vehicles and net zero economy," *Journal of Cleaner Production*, vol. 256, no. 120659, 2020.

[27] W. Lütkenhorst, T. Altenburg, A. Pegels, and G. Vidican, "Green Industrial Policy: Managing Transformation under Uncertainty," Deutsches Institut für Entwicklungspolitik, Tech. Rep., 2014.

[28] T. Foxon, G. Hammond, and P. Pearson, "Socio-technical transitions in UK electricity: part 2 – technologies and sustainability," *Proceedings of the Institution of Civil Engineers - Energy*, vol. 173, no. 3, pp. 123–136, 2020. [Online]. Available: https://www.icevirtuallibrary.com/doi/pdf/10.1680/jener.19.00076

[29] P. Johnstone and P. Kivimaa, "Multiple dimensions of disruption, energy transitions and industrial policy," *Energy Research and Social Science*, vol. 37, pp. 260–265, 2018.

[30] IRENA, "Renewable Power Generation Costs in 2017," IRENA International Renewable Energy Agency, Abu Dhabi, Tech. Rep., 2018. [Online]. Available: file:///C:/Users/Sean/Downloads/IRENA_2017_Power_Costs_2018(1).pdf

[31] Vivid Economics Limited, "Accelerated Electrification and the GB Electricity System," Imperial College London, London, Tech. Rep., 2019. [Online]. Available: https://www.theccc.org.uk/wp-content/uploads/2019/05/CCC-Accelerated-Electrification-Vivid-Economics-Imperial-1.pdf

[32] C. Zheng, J. Yuan, L. Zhu, Y. Zhang, and Q. Shao, "From digital to sustainable: A scientometric review of smart city literature between 1990 and 2019," *Journal of Cleaner Production*, vol. 258, no. 120689, 2020.

[33] A. Sodiq, A. A. Baloch, S. A. Khan, N. Sezer, S. Mahmoud, M. Jama, and A. Abdelaal, "Towards modern sustainable cities: Review of sustainability principles and trends," *Journal of Cleaner Production*, vol. 227, pp. 972–1001, 2019.

[34] F. Mosannenzadeh, A. Bisello, R. Vaccaro, V. D'Alonzo, G. W. Hunter, and D. Vettorato, "Smart energy city development: A story told by urban planners," *Cities*, vol. 64, pp. 54–65, 2017.

[35] S. J. Wang and P. Moriarty, "Energy savings from Smart Cities: A critical analysis," *Energy Procedia*, vol. 158, pp. 3271–3276, 2019.

[36] M. L. D. Silvestre, S. Favuzza, E. Riva Sanseverino, and G. Zizzo, "How Decarbonization, Digitalization and Decentralization are changing key power infrastructures," *Renewable and Sustainable Energy Reviews*, vol. 93, pp. 483–498, 2018.

[37] J. Z. Thellufsen, H. Lund, P. Sorknæs, P. A. Østergaard, M. Chang, D. Drysdale, S. Nielsen, S. R. Djørup, and K. Sperling, "Smart energy cities in a 100% renewable energy context," *Renewable and Sustainable Energy Reviews*, vol. 129, no. 109922, 2020.

[38] S. Marinova, S. Deetman, E. van der Voet, and V. Daioglou, "Global construction materials database and stock analysis of residential buildings between 1970-2050," *Journal of Cleaner Production*, vol. 247, no. 119146, 2020.

[39] R. Qiao and T. Liu, "Impact of building greening on building energy consumption: A quantitative computational approach," *Journal of Cleaner Production*, vol. 246, no. 119020, 2020.

[40] K. Paridari and L. Nordström, "Flexibility prediction, scheduling and control of aggregated TCLs," *Electric Power Systems Research*, vol. 178, no. 106004, 2020.

[41] O. Ibrahim, F. Fardoun, R. Younes, and H. Louahlia-Gualous, "Review of water-heating systems: General selection approach based on energy and environmental aspects," *Building and Environment*, vol. 72, pp. 259–286, 2014.

[42] L. Yang, H. Yan, and J. C. Lam, "Thermal comfort and building energy consumption implications - A review," *Applied Energy*, vol. 115, pp. 164–173, 2014.

[43] F. Ascione, N. Bianco, G. M. Mauro, D. F. Napolitano, and G. P. Vanoli, "Weather-data-based control of space heating operation via multi-objective optimization: Application to Italian residential buildings," *Applied Thermal Engineering*, vol. 163, no. 114384, 2019.

[44] S. Lin, D. Liu, F. Hu, F. Li, W. Dong, D. Li, and Y. Fu, "Grouping control strategy for aggregated thermostatically controlled loads," *Electric Power Systems Research*, vol. 171, pp. 97–104, 2019.

[45] S. Erdogan, S. Yıldırım, D. C. Yıldırım, and A. Gedikli, "The effects of innovation on sectoral carbon emissions: Evidence from G20 countries," *Journal of Environmental Management*, vol. 267, no. 110637, 2020.

[46] J. Kabayo, P. Marques, R. Garcia, and F. Freire, "Life-cycle sustainability assessment of key electricity generation systems in Portugal," *Energy*, vol. 176, pp. 131–142, 2019.

[47] H. T. Marcondes dos Santos and J. A. Perrella Balestieri, "Spatial analysis of sustainable development goals: A correlation between socioeconomic variables and electricity use," *Renewable and Sustainable Energy Reviews*, vol. 97, pp. 367–376, 2018.

[48] D. Kumar, H. D. Mathur, S. Bhanot, and R. C. Bansal, "Modeling and frequency control of community micro-grids under stochastic solar and wind sources," *Engineering Science and Technology, an International Journal*, 2020.

[49] R. Loisel and L. Lemiale, "Comparative energy scenarios: Solving the capacity sizing problem on the French Atlantic Island of Yeu," *Renewable and Sustainable Energy Reviews*, vol. 88, pp. 54–67, 2018.

[50] D. Curto, S. Favuzza, V. Franzitta, R. Musca, M. A. Navarro Navia, and G. Zizzo, "Evaluation of the optimal renewable electricity mix for Lampedusa island: The adoption of a technical and economical methodology," *Journal of Cleaner Production*, vol. 263, no. 121404, 2020.

[51] P. Cabrera, H. Lund, and J. A. Carta, "Smart renewable energy penetration strategies on islands: The case of Gran Canaria," *Energy*, vol. 162, pp. 421–443, 2018.

[52] S. Tindemans and G. Strbac, "Low-complexity control algorithm for decentralised demand response using thermostatic loads," in *Proceedings - 2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe, EEEIC/I and CPS Europe 2019*, 2019.

[53] C. Schellenberg, L. Dimache, and J. Lohan, "Grid-edge technology - Exploring the flexibility potential of a heat pump and thermal energy storage system," in *E3S Web of Conferences*, 2019.

[54] Y. Shen, Y. Li, Q. Zhang, Q. Shi, and F. Li, "State-shift priority based progressive load control of residential HVAC units for frequency regulation," *Electric Power Systems Research*, vol. 182, 2020.

[55] D. Croce, F. Giuliano, M. Bonomolo, G. Leone, R. Musca, and I. Tinnirello, "A decentralized load control architecture for smart energy consumption in small islands," *Sustainable Cities and Society*, vol. 53, 2020.

[56] C. Esposito, O. Tamburis, X. Su, and C. Choi, "Robust Decentralised Trust Management for the Internet of Things by Using Game Theory," *Information Processing and Management*, vol. 57, no. 102308, 2020. [Online]. Available: https://doi.org/10.1016/j.ipm.2020.102308

[57] B. McLellan, N. Florin, D. Giurco, Y. Kishita, K. Itaoka, and T. Tezuka, "Decentralised energy futures: The changing emissions reduction landscape," *Procedia CIRP*, vol. 29, pp. 138–143, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212827115000943

[58] S. Breukers, T. Crosbie, and L. Van Summeren, "Mind the gap when implementing technologies intended to reduce or shift energy consumption in blocks-of-buildings," *Energy and Environment*, vol. 31, no. 4, pp. 613–633, 2020.

[59] M. Ala-Juusela, T. Crosbie, and M. Hukkalainen, "Defining and operationalising the concept of an energy positive neighbourhood," *Energy Conversion and Management*, vol. 125, pp. 133–140, 2016.

[60] T. Crosbie, J. Broderick, M. Short, R. Charlesworth, and M. Dawood, "Demand response technology readiness levels for energy management in blocks of buildings," *Buildings*, vol. 8, no. 2, 2018.

[61] J. Zapata Riveros, M. Kubli, and S. Ulli-Beer, "Prosumer communities as strategic allies for electric utilities: Exploring future decentralization trends in Switzerland," *Energy Research and Social Science*, vol. 57, no. 101219, 2019.

[62] D. Brown, S. Hall, and M. E. Davis, "What is prosumerism for? Exploring the normative dimensions of decentralised energy transitions," *Energy Research and Social Science*, vol. 66, no. 101475, 2020.

[63] P. Martínez Fernández, I. Villalba Sanchís, V. Yepes, and R. Insa Franco, "A review of modelling and optimisation methods applied to railways energy consumption," *Journal of Cleaner Production*, vol. 222, pp. 153–162, 2019.

[64] H. Borhanazad, S. Mekhilef, V. Gounder Ganapathy, M. Modiri-Delshad, and A. Mirtaheri, "Optimization of micro-grid system using MOPSO," *Renewable Energy*, vol. 71, pp. 295–306, 2014.

[65] T. M. Skjølsvold, M. Ryghaug, and W. Throndsen, "European island imaginaries: Examining the actors, innovations, and renewable energy transitions of 8 islands," *Energy Research and Social Science*, vol. 65, no. 101491, 2020.

[66] L. Rodrigues, M. Gillott, J. A. Waldron, L. Cameron, R. Tubelo, R. Shipman, N. Ebbs, and C. Bradshaw-Smith, "User engagement in community energy schemes: A case study at the Trent Basin in Nottingham, UK," *Sustainable Cities and Society*, vol. 61, no. 102187, 2020.

[67] F. Fuentes González, A. H. van der Weijde, and E. Sauma, "The promotion of community energy projects in Chile and Scotland: An economic approach using biform games," *Energy Economics*, vol. 144, no. 111678, 2020.

[68] L. F. van Summeren, A. J. Wieczorek, G. J. Bombaerts, and G. P. Verbong, "Community energy meets smart grids: Reviewing goals, structure, and roles in Virtual Power Plants in Ireland, Belgium and the Netherlands," *Energy Research and Social Science*, vol. 63, no. 101415, 2020.

[69] M. B. Lindberg, J. Markard, and A. D. Andersen, "Policies, actors and sustainability transition pathways: A study of the EU's energy policy mix," *Research Policy*, vol. 48, no. 103668, 2019.

[70] E. Creamer, W. Eadson, B. van Veelen, A. Pinker, M. Tingey, T. Braunholtz-Speight, M. Markantoni, M. Foden, and M. Lacey-Barnacle, "Community energy: Entanglements of community, state, and private sector," *Geography Compass*, vol. 12, no. 7, pp. 1–16, 2018.

[71] H. Roby and S. Dibb, "Future pathways to mainstreaming community energy," *Energy Policy*, vol. 135, no. 111020, 2019.

[72] R. E. Bush, C. S. Bale, M. Powell, A. Gouldson, P. G. Taylor, and W. F. Gale, "The role of intermediaries in low carbon transitions – Empowering innovations to unlock district heating in the UK," *Journal of Cleaner Production*, vol. 148, pp. 137–147, 2017.

[73] O. Soyinka, K. W. M. Siu, T. Lawanson, and O. Adeniji, "Assessing smart infrastructure for sustainable urban development in the Lagos metropolis," *Journal of Urban Management*, vol. 5, no. 2, pp. 52–64, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2226585616300292

[74] UN Department of Economics and Social Affairs, "World Population Prospects - Population Division - United Nations." [Online]. Available: https://population.un.org/wpp/Graphs/Probabilistic/POP/TOT/900

[75] J. Freeman, E. Robinson, N. G. Beckman, D. Bird, J. A. Baggio, and J. M. Anderies, "The global ecology of human population density and interpreting changes in paleo-population density," *Journal of Archaeological Science*, vol. 120, no. 105168, 2020.

[76] V. Costantini and C. Martini, "The causality between energy consumption and economic growth: A multi-sectoral analysis using non-stationary cointegrated panel data," *Energy Economics*, vol. 32, no. 3, pp. 591–603, 2010.

[77] I. Ozturk, "A literature survey on energy-growth nexus," *Energy Policy*, vol. 38, pp. 340–349, 2010.

[78] M. Shahbaz, S. Sarwar, W. Chen, and M. N. Malik, "Dynamics of electricity consumption, oil price and economic growth: Global perspective," *Energy Policy*, vol. 108, pp. 259–270, 2017.

[79] F. Faisal, T. Tursoy, and O. Ercantan, "The relationship between energy consumption and economic growth: Evidence from non-Granger causality test," in *Procedia Computer Science*, vol. 120. Elsevier B.V., 2017, pp. 671–675.

[80] P. K. Narayan and S. Popp, "The energy consumption-real GDP nexus revisited: Empirical evidence from 93 countries," *Economic Modelling*, vol. 29, pp. 303–308, 2012.

[81] T. Zachariadis, "Exploring the relationship between energy use and economic growth with bivariate models: New evidence from G-7 countries," *Energy Economics*, vol. 29, no. 6, pp. 1233–1253, 2007.

[82] S. Śmiech and M. Papiez, "Energy consumption and economic growth in the light of meeting the targets of energy policy in the EU: The bootstrap panel Granger causality approach," *Energy Policy*, vol. 71, pp. 118–129, 2014.

[83] L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," *Energy and Buildings*, vol. 40, no. 3, pp. 394–398, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778807001016

[84] Y. Li, M. Han, S. Liu, and G. Chen, "Energy consumption and greenhouse gas emissions by buildings: A multi-scale perspective," *Building and Environment*, vol. 151, pp. 240–250, 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360132318306917

[85] M. C. Sing, P. E. Love, and H. J. Liu, "Rehabilitation of existing building stock: A system dynamics model to support policy development," *Cities*, vol. 87, pp. 142–152, 2019. [Online]. Available: www.elsevier.com/locate/cities

[86] D. B. Avancini, J. J. Rodrigues, S. G. Martins, R. A. Rabêlo, J. Al-Muhtadi, and P. Solic, "Energy meters evolution in smart grids: A review," *Journal of Cleaner Production*, vol. 217, pp. 702–715, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959652619302501

[87] J. Malinauskaite, H. Jouhara, L. Ahmad, M. Milani, L. Montorsi, and M. Venturelli, "Energy efficiency in industry: EU and national policies in Italy and the UK," *Energy*, vol. 172, pp. 255–269, 2019.

[88] O. Peia and K. Roszbach, "Finance and growth: Time series evidence on causality," *Journal of Financial Stability*, vol. 19, pp. 105–118, 2013.

[89] B. Ghosh, B. Basu, and M. O'Mahony, "Multivariate short-term traffic flow forecasting using time-series analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 246–254, 2009.

[90] R. Fildes, S. Ma, and S. Kolassa, "Retail forecasting: Research and practice," *International Journal of Forecasting*, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016920701930192X

[91] K. Nouira and A. Trabelsi, "Time series analysis and outlier detection in intensive care data," in *International Conference on Signal Processing Proceedings, ICSP*, vol. 4. Institute of Electrical and Electronics Engineers Inc., 2006.

[92] T. Ahmad and H. Chen, "Utility companies strategy for short-term energy demand forecasting using machine learning based models," *Sustainable Cities and Society*, vol. 39, pp. 401–417, 2018. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S221067071731096X

[93] S. Muzaffar and A. Afshari, "Short-Term Load Forecasts Using LSTM Networks," *Energy Procedia*, vol. 158, pp. 2922–2927, 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1876610219310008

[94] C. Robinson, B. Dilkina, J. Hubbs, W. Zhang, S. Guhathakurta, M. A. Brown, and R. M. Pendyala, "Machine learning approaches for estimating commercial building energy consumption," *Applied Energy*, vol. 208, pp. 889–904, 2017.

[95] A. González-Vidal, F. Jiménez, and A. F. Gómez-Skarmeta, "A methodology for energy multivariate time series forecasting in smart buildings based on feature selection," *Energy and Buildings*, vol. 196, pp. 71–82, 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378778818338775

[96] R. Nepal and N. Paija, "A multivariate time series analysis of energy consumption," *Economic Modelling*, vol. 77, pp. 164–173, 2019.

[97] M. Chayama and Y. Hirata, "When univariate model-free time series prediction is better than multivariate," *Physics Letters A*, vol. 380, no. 31-32, pp. 2359–2365, 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0375960116302195

[98] A. Inoue, L. Jin, and B. Rossi, "Rolling window selection for out-of-sample forecasting with time-varying parameters," *Journal of Econometrics*, vol. 196, pp. 55–67, 2017.

[99] J. W. Galbraith, "Content horizons for univariate time-series forecasts," *International Journal of Forecasting*, vol. 19, pp. 43–45, 2003.

[100] I. Naim, T. Mahara, and A. R. Idrisi, "Effective Short-Term Forecasting for Daily Time Series with Complex Seasonal Patterns," in *Procedia Computer Science*, 2018, pp. 1832–1841.

[101] J. Gardner Everette S, "Exponential smoothing: The state of the art—Part II," *International Journal of Forecasting*, vol. 22, pp. 637–666, 2006.

[102] V. S. Ediger and S. Akar, "ARIMA forecasting of primary energy demand by fuel in Turkey," *Energy Policy*, vol. 35, pp. 1707–1708, 2007.

[103] S. Noureen, S. Atique, V. Roy, and S. Bayne, "Analysis and application of seasonal ARIMA model in Energy Demand Forecasting: A case study of small scale agricultural load," in *Midwest Symposium on Circuits and Systems*, 2019, pp. 521–521.

[104] J. W. Taylor, "Short-term load forecasting with exponentially weighted methods," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 458–464, 2012.

[105] E. Arsenault, J. T. Bernard, C. W. Carr, and E. Genest-Laplante, "A total energy demand model of Québec. Forecasting properties," *Energy Economics*, vol. 17, no. 2, pp. 163–171, 1995.

[106] J. S. Chou and D. S. Tran, "Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders," *Energy*, vol. 165, pp. 709–726, 2018.

[107] G. P. Herrera, M. Constantino, B. M. Tabak, H. Pistori, J. J. Su, and A. Naranpanawa, "Long-term forecast of energy commodities price using machine learning," *Energy*, vol. 179, pp. 214–221, 2019.

[108] H. Demolli, A. S. Dokuz, A. Ecemis, and M. Gokcek, "Wind power forecasting based on daily wind speed data using machine learning algorithms," *Energy Conversion and Management*, vol. 198, no. 111823, 2019.

[109] S. E. Schaeffer and S. V. Rodriguez Sanchez, "Forecasting client retention — A machine-learning approach," *Journal of Retailing and Consumer Services*, vol. 52, no. 101918, 2020.

[110] N. Somu, G. R. M R, and K. Ramamritham, "A hybrid model for building energy consumption forecasting using long short term memory networks," *Applied Energy*, vol. 261, no. 114131, 2020.

[111] A. Yang, W. Li, and X. Yang, "Short-term electricity load forecasting based on feature selection and Least Squares Support Vector Machines," *Knowledge-Based Systems*, vol. 163, pp. 159–173, 2019.

[112] H. J. Sadaei, P. C. de Lima e Silva, F. G. Guimarães, and M. H. Lee, "Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series," *Energy*, vol. 175, pp. 365–377, 2019.

[113] M. S. Al-Musaylh, R. C. Deo, J. F. Adamowski, and Y. Li, "Short-term electricity demand forecasting with MARS, SVR and ARIMA models using aggregated demand data in Queensland, Australia," *Advanced Engineering Informatics*, vol. 35, pp. 1–16, 2018.

[114] C. J. Willmott, S. M. Robeson, and K. Matsuura, "A refined index of model performance," *International Journal of Climatology*, vol. 32, no. 13, pp. 2088–2094, 2012.

[115] M. Sherris and C. N. Njenga, "Modeling Mortality with a Bayesian Vector Autoregression," *SSRN Electronic Journal*, vol. 94, pp. 40–57, 2012.

[116] M. Duran Toksari, "Ant colony optimization approach to estimate energy demand of Turkey," *Energy Policy*, vol. 35, pp. 3984–3990, 2007.

[117] O. C. Ozerdem, E. O. Olaniyi, and O. K. Oyedotun, "Short term load forecasting using particle swarm optimization neural network," in *Procedia Computer Science*, 2017, pp. 382–393.

[118] M. El-Telbany and F. El-Karmi, "Short-term forecasting of Jordanian electricity demand using particle swarm optimization," *Electric Power Systems Research*, vol. 78, pp. 425–433, 2008.

[119] S. Sumathi and S. N. Sivanandam, *Introduction to Data Mining and its Applications*. Springer-Verlag Berlin Heidelberg, 2006.

[120] F. Martínez-Álvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar Ruiz, "Energy time series forecasting based on pattern sequence similarity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 8, 2011.

[121] I. Manojlović, G. Švenda, A. Erdeljan, and M. Gavrić, "Time series grouping algorithm for load pattern recognition," *Computers in Industry*, vol. 111, pp. 140–147, 2019.

[122] O. O. Aremu, D. Hyland-Wood, and P. R. McAree, "A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data," *Reliability Engineering and System Safety*, vol. 195, no. 106706, 2020.

[123] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[124] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[125] Q. Wang and V. Megalooikonomou, "A dimensionality reduction technique for efficient time series similarity analysis," *Information Systems*, vol. 33, no. 1, pp. 115–132, 2008.

[126] J. S. Arora, "Introduction to Design Optimization," in *Introduction to Optimum Design*. Elsevier, 2017, ch. 1, pp. 3–18.

[127] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *In: Proceedings of the IEEE international joint conference on neural networks*, vol. 4, no. 6, pp. 1942–1948, 1995.

[128] S. Fong, S. Deb, X. S. Yang, and J. Li, "Feature selection in life science classification: Metaheuristic swarm search," *IT Professional*, vol. 16, no. 4, pp. 24–29, 2014.

[129] Y. T. Zhang and T. Zhi, "Optimization of least-squares localization algorithm in WSN," in *26th Chinese Control and Decision Conference, CCDC 2014*. IEEE Computer Society, 2014, pp. 2198–2202.

[130] X. Dastile, T. Celik, and M. Potsane, "Statistical and machine learning models in credit scoring: A systematic literature survey," *Applied Soft Computing Journal*, vol. 91, no. 106263, 2020.

[131] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018.

[132] C. F. Wood, "Review of Design Optimization Techniques," *IEEE Transactions on Systems Science and Cybernetics*, vol. 1, no. 1, pp. 14–20, 1965.

[133] P. P. Selvam and R. Narayanan, "Random Restart Local Search Optimization technique for sustainable energy-generating induction machine," *Computers and Electrical Engineering*, vol. 73, pp. 268–278, 2019.

[134] T. Liu, H. Yu, H. Guo, Y. Qin, and Y. Zou, "Online Energy Management for Multimode Plug-In Hybrid Electric Vehicles," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4352–4361, 7 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8532280/

[135] A. Zakaria, F. B. Ismail, M. S. Lipu, and M. A. Hannan, "Uncertainty models for stochastic optimization in renewable energy applications," *Renewable Energy*, vol. 145, pp. 1543–1571, 2020.

[136] O. Altıntas, B. Okten, O. Karsu, and A. S. Kocaman, "Bi-objective optimization of a grid-connected decentralized energy system," *International Journal of Energy Research*, vol. 42, no. 2, pp. 447–465, 2018. [Online]. Available: http://doi.wiley.com/10.1002/er.3813

[137] A. Downward, O. Dowson, and R. Baucke, "Stochastic dual dynamic programming with stagewise-dependent objective uncertainty," *Operations Research Letters*, vol. 48, no. 1, pp. 33–39, 2020.

[138] S. W. Wallace and S. E. Fleten, "Stochastic Programming Models in Energy," *Handbooks in Operations Research and Management Science*, vol. 10, pp. 637–677, 2003.

[139] M. V. Pereira and L. M. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical Programming*, vol. 52, pp. 359–375, 1991.

[140] N. Hashempour, R. Taherkhani, and M. Mahdikhani, "Energy performance optimization of existing buildings: A literature review," *Sustainable Cities and Society*, vol. 54, no. 101967, 2020.

[141] M. H. Arslan, M. Ceylan, M. Y. Kaltakci, Y. Ozbay, and F. G. Gulay, "Prediction of force reduction factor (R) of prefabricated industrial buildings using neural networks," *Structural Engineering and Mechanics*, vol. 27, no. 2, pp. 117–134, 2007.

[142] M. H. Arslan, "An evaluation of effective design parameters on earthquake performance of RC buildings using neural networks," *Engineering Structures*, vol. 32, pp. 1888–1898, 2010.

[143] S. R. Mohandes, X. Zhang, and A. Mahdiyar, "A comprehensive review on the application of artificial neural networks in building energy analysis," *Neurocomputing*, vol. 340, pp. 55–75, 2019.

[144] T. Olofsson and S. Andersson, "Overall heat loss coefficient and domestic energy gain factor for single-family buildings," *Building and Environment*, vol. 37, no. 11, pp. 1019–1026, 2002.

[145] M. Macarulla, M. Casals, N. Forcada, and M. Gangolells, "Implementation of predictive control in a commercial building energy management system using neural networks," *Energy and Buildings*, vol. 151, pp. 511–519, 2017.

[146] X. Yuan, C. Chen, M. Jiang, and Y. Yuan, "Prediction interval of wind power using parameter optimized Beta distribution based LSTM model," *Applied Soft Computing Journal*, vol. 82, no. 105550, 2019.

[147] X. Song, Y. Liu, L. Xue, J. Wang, J. Zhang, J. Wang, L. Jiang, and Z. Cheng, "Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model," *Journal of Petroleum Science and Engineering*, vol. 186, no. 106682, 2020.

[148] P. Jiang, H. Yang, and J. Heng, "A hybrid forecasting system based on fuzzy time series and multi-objective optimization for wind speed forecasting," *Applied Energy*, vol. 235, pp. 786–801, 2019.

[149] Z. Palmowski and A. Sidorowicz, "An application of dynamic programming to assign pressing tanks at wineries," *European Journal of Operational Research*, vol. 287, pp. 293–305, 2020.

[150] H. Pombeiro, M. J. Machado, and C. Silva, "Dynamic programming and genetic algorithms to control an HVAC system: Maximizing thermal comfort and minimizing cost with PV production and storage," *Sustainable Cities and Society*, vol. 34, pp. 228–238, 2017.

[151] J. Niu, Z. Tian, Y. Lu, and H. Zhao, "Flexible dispatch of a building energy system using building thermal storage and battery energy storage," *Applied Energy*, vol. 243, pp. 274–287, 2019.

[152] M. Short, S. Rodriguez, R. Charlesworth, T. Crosbie, and N. Dawood, "Optimal dispatch of aggregated HVAC units for demand response: An industry 4.0 approach," *Energies*, vol. 12, no. 4320, 2019.

[153] G. Costa Silva, E. E. Carvalho, and W. M. Caminhas, "An artificial immune systems approach to Case-based Reasoning applied to fault detection and diagnosis," *Expert Systems with Applications*, vol. 140, no. 112906, 2020.

[154] R. Faia, T. Pinto, O. Abrishambaf, F. Fernandes, Z. Vale, and J. M. Corchado, "Case based reasoning with expert system and swarm intelligence to determine energy reduction in buildings energy management," *Energy and Buildings*, vol. 155, pp. 269–281, 2017.

[155] N. Delgarm, B. Sajadi, and S. Delgarm, "Multi-objective optimization of building energy performance and indoor thermal comfort: A new method using artificial bee colony (ABC)," *Energy and Buildings*, vol. 131, pp. 42–53, 2016.

[156] A. Ghahramani, S. A. Karvigh, and B. Becerik-Gerber, "HVAC system energy optimization using an adaptive hybrid metaheuristic," *Energy and Buildings*, vol. 152, pp. 149–161, 2017.

[157] S. F. Mussati, K. V. Gernaey, T. Morosuk, and M. C. Mussati, "NLP modeling for the optimization of LiBr-H2O absorption refrigeration systems with exergy loss rate, heat transfer area, and cost as single objective functions," *Energy Conversion and Management*, vol. 127, pp. 526–544, 11 2016.

[158] J. Granat and F. Guerriero, "The interactive analysis of the multicriteria shortest path problem by the reference point method," *European Journal of Operational Research*, vol. 151, pp. 109–118, 2003.

[159] C. Tung Tung and K. Lin Chew, "A multicriteria Pareto-optimal path algorithm," *European Journal of Operational Research*, vol. 62, no. 2, pp. 203–209, 1992.

[160] V. Kapsalis and L. Hadellis, "Optimal operation scheduling of electric water heaters under dynamic pricing," *Sustainable Cities and Society*, vol. 31, pp. 109–121, 2017.

[161] N. Kampelis, E. Tsekeri, D. Kolokotsa, K. Kalaitzakis, D. Isidori, and C. Cristalli, "Development of demand response energy management optimization at building and district levels using genetic algorithm and artificial neural network modelling power predictions," *Energies*, vol. 11, no. 3012, 2018.

[162] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[163] H. Bahlawan, M. Morini, M. Pinelli, and P. R. Spina, "Dynamic programming based methodology for the optimization of the sizing and operation of hybrid energy plants," *Applied Thermal Engineering*, vol. 160, no. 113967, 2019.

[164] C.-J. Dalgaard and H. Strulik, "Energy distribution and economic growth," *Resource and Energy Economics*, vol. 33, no. 4, pp. 782–797, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0928765511000273

[165] International Energy Agency (IEA), "World Energy Outlook 2017," 2017.

[166] S. Verbeke and A. Audenaert, "Thermal inertia in buildings: A review of impacts across climate and building use," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 2300–2318, 2017.

[167] A. Eichler, G. Darivianakis, and J. Lygeros, "Humans in the Loop: A Stochastic Predictive Approach to Building Energy Management in the Presence of Unpredictable Users," *IFAC Proceedings Volumes*, vol. 50, no. 2014, pp. 15 036–15 041, 2017. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S2405896317331063

[168] A. Reilly and O. Kinnane, "The impact of thermal mass on building energy consumption," *Applied Energy*, vol. 198, pp. 108–121, 2017.

[169] E. Rodrigues, R. Godina, E. Pouresmaeil, J. Ferreira, and J. Catalão, "Domestic appliances energy optimization with model predictive control," *Energy Conversion and Management*, vol. 142, pp. 402–413, 2017.

[170] M. Dreidy, H. Mokhlis, and S. Mekhilef, "Inertia response and frequency control techniques for renewable energy sources: A review," *Renewable and Sustainable Energy Reviews*, vol. 69, pp. 144–155, 2017.

[171] E. Kremers, J. M. González De Durana, and O. Barambones, "Emergent synchronisation properties of a refrigerator demand side management system," *Applied Energy*, vol. 101, pp. 709–717, 2013.

[172] D. S. Callaway, "Tapping the energy storage potential in electric loads to deliver load following and regulation, with application to wind energy," *Energy Conversion and Management*, vol. 50, no. 5, pp. 1389–1400, 2009.

[173] C. He and H. Wang, "Research on primary frequency control strategy based on DFIG," in *2012 IEEE Innovative Smart Grid Technologies - Asia, ISGT Asia 2012*, 2012.

[174] C. Yang, J. Yao, W. Lou, and S. Xie, "On Demand Response Management Performance Optimization for Microgrids under Imperfect Communication Constraints," *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 881–893, 2017.

[175] S. Williams, M. Short, and T. Crosbie, "Evaluating the role of building thermal inertia for the provision of decentralised demand side primary electrical frequency regulation services," in *SusTEM2017*, 2017.

[176] N. Grid, E. Transmission, E. Act, G. Britain, and P. Act, "The Grid Code Issue 5," National Grid, Tech. Rep., 2020.

[177] National Grid, "Historic frequency data," 2020. [Online]. Available: https://www.nationalgrideso.com/balancing-services/frequency-response-services/historic-frequency-data

[178] S. Homan and S. Brown, "An analysis of frequency events in Great Britain," *Energy Reports*, vol. 6, pp. 63–69, 2020.

[179] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03*, 2003, pp. 2–11.

[180] H. Bevrani and T. Hiyama, "Robust decentralised PI based LFC design for time delay power systems," *Energy Conversion and Management*, vol. 49, no. 2, pp. 193–204, 2008.

[181] Q. Shi, F. Li, Q. Hu, and Z. Wang, "Dynamic demand control for system frequency regulation: Concept review, algorithm comparison, and future vision," *Electric Power Systems Research*, vol. 154, pp. 75–87, 2018.

[182] R. Sharma and G. Dhole, "Least Error Squares Approach: A Practical Method for Power System Frequency and Amplitude Estimation," *Procedia Technology*, vol. 25, pp. 710–717, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212017316305114

[183] Atmel Corporation, "Atmel ATmega2560 Datasheet," 2014. [Online]. Available: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

[184] British Stanardards Online, "BS ISO 8601-2:2019," Bsi, Tech. Rep., 2019. [Online]. Available: https://bsol-bsigroup-com.ezproxy.tees.ac.uk/Bibliographic/BibliographicInfoData/000000000030339390

[185] MathWorks, "MATLAB and Statistics Toolbox Release 2018b," Massachusetts, 2018.

[186] S. Nag and N. Philip, "Application of neural networks to automatic load frequency control," in *Control, Instrumentation, Energy and Communication (CIEC), 2014 International Conference on*, 2014, pp. 345–350.

[187] K. U. Rao, "Load Frequency Control," in *Computer Techniques and Models in Power Systems*, 2nd ed. New Delhi: I.K. International Publishing House, 2015, ch. 13, pp. 523–557.

[188] S. Wang and R. Tang, "Supply-based feedback control strategy of air-conditioning systems for direct load control of buildings responding to urgent requests of smart grids," *Applied Energy*, vol. 201, pp. 419–432, 2017.

[189] E. Mathews, P. Richards, and C. Lombard, "A first-order thermal model for building design," *Energy and Buildings*, vol. 21, no. 2, pp. 133–145, 1994.

[190] R. F. Yan, T. K. Saha, N. Modi, N. A. Masood, and M. Mosadeghy, "The combined effects of high penetration of wind and PV on power system frequency response," *Applied Energy*, vol. 145, no. 145, pp. 320–330, 2015.

[191] Feedback Instruments, "Process Trainer PT326," 1982. [Online]. Available: https://eclass.upatras.gr/modules/document/file.php/EE817/thermic_PT_326_manual.pdf

[192] T. E. Olsen and B. E. Bialkowski, "Lambda Tuning as a Promising Controller Tuning Method for the Refinery," *AIChE Spring National Meeting*, vol. 42, 2002.

[193] D. E. Rivera, M. Morari, and S. Skogestad, "Internal model control: PID controller design," *Ind. Eng. Chem. Process Des. Dev*, vol. 25, pp. 252–265, 1986. [Online]. Available: http://pubs.acs.org/doi/pdf/10.1021/i200032a041

[194] F. B. Larson R., *Elementary statistics: picturing the world*, 5th ed. Prentice Hall, 2012.

[195] E. Kamir, F. Waldner, and Z. Hochman, "Estimating wheat yields in Australia using climate records, satellite image time series and machine learning methods," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 160, pp. 124–135, 2020.

[196] D. A. Wood, "German solar power generation data mining and prediction with transparent open box learning network integrating weather, environmental and market variables," *Energy Conversion and Management*, vol. 196, pp. 354–369, 2019.

[197] T. Alsuliman, D. Humaidan, and L. Sliman, "Machine learning and artificial intelligence in the service of medicine: Necessity or potentiality?" *Current Research in Translational Medicine*, 2020.

[198] A. Jastrzebska, "Time series classification through visual pattern recognition," *Journal of King Saud University - Computer and Information Sciences*, 2020.

[199] S. Saleti and S. R.B.V., "A MapReduce solution for incremental mining of sequential patterns from big data," *Expert Systems with Applications*, vol. 133, pp. 109–125, 2019.

[200] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, 2000.

[201] L. Kahane, "The Least-Squares Estimation Method," in *Regression Basics*. Thousand Oaks: SAGE Publications Inc, 2014, ch. 2, pp. 17–24.

[202] S. Makridakis and M. Hibon, "The M3-competition: Results, conclusions and implications," *International Journal of Forecasting*, vol. 16, pp. 451–476, 2000.

[203] E. Parzen, "ARARMA models for time series analysis and forecasting," *Journal of Forecasting*, vol. 1, no. 1, pp. 67–82, 1982.

[204] J. Bedi and D. Toshniwal, "Deep learning framework to forecast electricity demand," *Applied Energy*, vol. 238, pp. 1312–1326, 2019.

[205] S. H. Tindemans, V. Trovato, and G. Strbac, "Decentralized Control of Thermostatic Loads for Flexible Demand Response," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1685–1700, 2015.

[206] S. Uski, E. Rinne, and J. Sarsama, "Microgrid as a cost-effective alternative to rural network underground cabling for adequate reliability," *Energies*, vol. 11, p. 8, 2018.

[207] P. Bradley, A. Coke, and M. Leach, "Financial incentive approaches for reducing peak electricity demand, experience from pilot trials with a UK energy provider," *Energy Policy*, vol. 98, pp. 108–120, 2016.

[208] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 369–271, 1959.

[209] S. Williams and M. Short, "Electricity Demand Forecasting in Decentralised Demand Side Response for Blocks of Building," in *International Conference on Energy and Sustainable Futures*, 2019.

[210] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[211] Q. Bi, W. J. Cai, Q. G. Wang, C. C. Hang, Eng-Lock Lee, Y. Sun, K. D. Liu, Y. Zhang, and B. Zou, "Advanced controller auto-tuning and its application in HVAC systems," *Control Engineering Practice*, vol. 8, pp. 633–644, 2000.

[212] G. Happle, J. A. Fonseca, and A. Schlueter, "A review on occupant behavior in urban building energy models," *Energy and Buildings*, no. 174, pp. 276–292, 2018.

[213] R. Likert, "A Technique for the Measurement of Attitudes," *Archieves of Psychology*, vol. 22, pp. 5–53, 1932.

[214] A. Joshi, S. Kale, S. Chandel, and D. Pal, "Likert Scale: Explored and Explained," *British Journal of Applied Science & Technology*, vol. 7, no. 4, pp. 396–403, 2015.

[215] S. Williams and M. Short, "Electricity demand forecasting for decentralised energy management," *Energy and Built Environment*, vol. 1, no. 2, pp. 178–186, 2020.

[216] Green Energy UK, "Green Energy UK," 2020. [Online]. Available: https://www.greenenergyuk.com/

[217] M. M. Lehman and J. F. Ramil, "Rules and Tools for Software Evolution Planning and Management," *Annals of Software Engineering*, vol. 11, no. 1, pp. 15–44, 2001.

[218] P. De Wilde, "The gap between predicted and measured energy performance of buildings: A framework for investigation," *Automation in Construction*, vol. 41, pp. 40–49, 2014.

[219] P. X. Zou, D. Wagle, and M. Alam, "Strategies for minimizing building energy performance gaps between the design intend and the reality," *Energy and Buildings*, vol. 191, pp. 31–41, 2019.

[220] C. Sun, R. Zhang, S. Sharples, Y. Han, and H. Zhang, "Thermal comfort, occupant control behaviour and performance gap – A study of office buildings in north-east China using data mining," *Building and Environment*, vol. 149, pp. 305–321, 2019.

[221] M. Pritoni, K. Salmon, A. Sanguinetti, J. Morejohn, and M. Modera, "Occupant thermal feedback for improved efficiency in university buildings," *Energy and Buildings*, vol. 144, pp. 241–250, 2017.

[222] ASHRAE, "Standard 55 – Thermal Environmental Conditions for Human Occupancy." [Online]. Available: https://www.ashrae.org/technical-resources/bookstore/standard-55-thermal-environmental-conditions-for-human-occupancy

[223] British Stanardards Online, "BS ISO 7730:2005," Bsi, Tech. Rep., 2016. [Online]. Available: https://bsol-bsigroup-com.ezproxy.tees.ac.uk/Bibliographic/BibliographicInfoData/000000000030046382

[224] A. Figueiredo, J. Kämpf, R. Vicente, R. Oliveira, and T. Silva, "Comparison between monitored and simulated data using evolutionary algorithms: Reducing the performance gap in dynamic building simulation," *Journal of Building Engineering*, vol. 17, pp. 96–106, 2018.

[225] J. Ngarambe, G. Y. Yun, and M. Santamouris, "The use of artificial intelligence (AI) methods in the prediction of thermal comfort in buildings: energy implications of AI-based thermal comfort controls," *Energy and Buildings*, vol. 211, no. 109807, 2020.

[226] Z. Wang, R. de Dear, M. Luo, B. Lin, Y. He, A. Ghahramani, and Y. Zhu, "Individual difference in thermal comfort: A literature review," *Building and Environment*, vol. 138, pp. 181–193, 2018.

[227] Z. Wang, H. Zhang, Y. He, M. Luo, Z. Li, T. Hong, and B. Lin, "Revisiting individual and group differences in thermal comfort based on ASHRAE database," *Energy and Buildings*, vol. 219, no. 110017, 2020.

[228] R. Hamzeh and X. Xu, "Technology selection methods and applications in manufacturing A review from 1990 to 2017," *Computers and Industrial Engineering*, vol. 138, no. 106123, 2019.

[229] N. Fraser, "Ten things we've learned from Blockly," in *Proceedings - 2015 IEEE Blocks and Beyond Workshop, Blocks and Beyond 2015*, 2015, pp. 49–50.

[230] A. Begel and E. Klopfer, "StarLogo TNG: An Introduction to Game Development," *Journal of E-Learning*, vol. 53, p. 146, 2007. [Online]. Available: http://langwidge.com/starlogo.pdf

[231] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, 2010.

[232] E. L. Eriksson and E. M. A. Gray, "Optimization of renewable hybrid energy systems – A multi-objective approach," *Renewable Energy*, vol. 133, pp. 971–999, 2019.

[233] H. Wang, W. Mao, and L. Eriksson, "A Three-Dimensional Dijkstra's algorithm for multi-objective ship voyage optimization," *Ocean Engineering*, vol. 186, no. 106131, 2019.

[234] Y. D. Rosita, E. E. Rosyida, and M. A. Rudiyanto, "Implementation of dijkstra algorithm and multi-criteria decision-making for optimal route distribution," *Procedia Computer Science*, vol. 161, pp. 378–385, 2019.

[235] L. Shen, H. Shao, T. Wu, W. H. Lam, and E. C. Zhu, "An energy-efficient reliable path finding algorithm for stochastic road networks with electric vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 450–473, 2019.

[236] V. Kapsalis, G. Safouri, and L. Hadellis, "Cost/comfort-oriented optimization algorithm for operation scheduling of electric water heaters under dynamic pricing," *Journal of Cleaner Production*, vol. 198, pp. 1053–1065, 2018.

[237] A. Kusiak, G. Xu, and F. Tang, "Optimization of an HVAC system with a strength multi-objective particle-swarm algorithm," *Energy*, vol. 36, pp. 5935–5943, 2011.

[238] Y. Wang, Y. Ma, F. Song, Y. Ma, C. Qi, F. Huang, J. Xing, and F. Zhang, "Economic and efficient multi-objective operation optimization of integrated energy system considering electro-thermal demand response," *Energy*, vol. 205, no. 118022, 2020. [Online]. Available: https://doi.org/10.1016/j.energy.2020.118022

[239] S. Williams, M. Short, T. Crosbie, and M. Shadman-Pajouh, "Decentralised Informatics, Optimisation and Control Framework for Evolving Demand Response Services," *Energies*, vol. 13, no. 16, p. 4191, 2020.

[240] European Commission, "EU Energy System Integration Strategy," 2020. [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/FS_20_1295

[241] V. Margaras, "Islands of the EU: Taking account of their specific needs in EU policy," European Parliment, Tech. Rep., 2016.

[242] Weather Underground, "Weather History & Data Archive," 2020. [Online]. Available: https://www.wunderground.com/history

Appendix A

Frequency Measurement Design and Implementation

Appendix Contents

A.1 Catalogue of parts

Table A.1
Catalogue of parts

Item	Part ID	Item	Remark	Cost (£)
1	AD1	Arduino Mega2560 R3 Code: N31KU Link: http://www.maplin.co.uk		29.99
2	AD2	ProtoShield Expansion Full Kit Code: SKU USTPSEKFr1 Link: https://www.steelpuppet.com	Arduino Mega2560 Screw Terminal Expansion	15.79
3	ADS1	Arduino Ethernet Shield R3 with microSD card slot Code: N33KU Link: http://www.maplin.co.uk		19.99
4	ADS1_1	Lithium Coin Cell CR1220 3V Battery Code: ZB77J Link: http://www.maplin.co.uk		2.70
5	ADS2	RS232 Shield V2 Code: DEV-13029 ROHS Link: https://www.sparkfun.com		7.94
6	ADS3	Adafruit Ultimate GPS Breakout 66 Channel MTK3339 Code: RB-Ada-53 Link: http://www.robotshop.com/uk/		31.96
7	ADS3_1	3V GPS Antenna Magnetic Mount SMA Code: RB-Spa-135 Link: http://www.robotshop.com/uk/		10.36
8	ADS3_2	Interface Cable SMA to uFL Code: RB-Dfr-239		3.01

continued ...

... continued

Item	Part ID	Item	Remark	Cost (£)
		Link: http://www.robotshop.com/uk/		
9	ABR	Full Bridge Rectifier KBPC1005 2A Bridge	Max Volt Drop of 1.3V per diode	0.79
		Code: AQ98G	Case Style: B3	
		Link: http://www.maplin.co.uk	Signal Conditioning Circuit Item	
10	D1	Indicator LED, 12VDC, 6mm, wired, BLUE/BrCr	GPS Fix	3.04
		Item No: APM Q6P3C B12E	Panel ID: 6	
		Link: http://www.reichelt.com	Mounting hole dia: 6mm	
11	D2	Indicator LED, 12VDC, 6mm, wired, YELLOW/BrC	Write to on-board micro SD Card	3.04
		Item No: APM Q6P3C Y12E	Panel ID: 3	
		Link: http://www.reichelt.com	Mounting hole dia: 6mm	
12	D3	Indicator LED, 12VDC, 6mm, wired, RED/BrC	Event – Statutory Threshold	3.04
		Item No: APM Q6P3C R12E	Panel ID: 4	
		Link: http://www.reichelt.com	Mounting hole dia: 6mm	
13	D4	Indicator LED, 12VDC, 6mm, wired, GREEN/BrC	Event – Operational Threshold	3.04
		Item No: APM Q6P3C G12E	Panel ID: 5	
		Link: http://www.reichelt.com	Mounting hole dia: 6mm	
14	LCD1	LCD Display 16x2		7.49
		Model: LCD16x2WD		
		Link: http://www.hobbytronics.co.uk	Mounting hole dia: 6mm	
15	Q1	2N3904 NPN Transistor	Silicon Transistor in a T092 Style Case	0.19
		Code: QR40T	Signal Conditioning Circuit Item	
		Link: http://www.maplin.co.uk		
16	R1	Resistor 10k ohm	Metal File 0.6W	0.09
		Code: M10K		
		Link: http://www.maplin.co.uk		
17	R2	Resistor 10k ohm	Metal File 0.6W	0.09
		Code: M10K		

continued ...

... continued

Item	Part ID	Item	Remark	Cost (£)
		Link: http://www.maplin.co.uk		
18	R3	Resistor 22k ohm	Metal File 0.6W	0.09
		Code: M22K		
		Link: http://www.maplin.co.uk		
19	R7	Resistor 10k ohm	Metal File 0.6W	0.09
		Code: M10K	Signal Conditioning Circuit Item	
		Link: http://www.maplin.co.uk		
20	R8	Resistor 10k ohm	Metal File 0.6W	0.09
		Code: M10K	Signal Conditioning Circuit Item	
		Link: http://www.maplin.co.uk		
21	R9	Resistor 10k ohm	Metal File 0.6W	0.09
		Code: M10K	Signal Conditioning Circuit Item	
		Link: http://www.maplin.co.uk		
22	RV1	Variable resistor 10k ohm	LCD back light brightness control	-
		RS Stock No. 729-3555	Combined with Part ID: SW3-1	
		Link: http://uk.rs-online.com		
23	SW1	Push button switch SPST	Screen Cycle	1.86
		RS Stock No. 734-6827	Panel ID: 2	
		Mfr. Part No. PBL-A2	Mounting hole dia. 8mm	
		Link: http://uk.rs-online.com		
24	SW2	Push button switch SPST	Event Reset	1.86
		RS Stock No. 734-6827	Panel ID: 8	
		Mfr. Part No. PBL-A2	Mounting hole dia. 8mm	
		Link: http://uk.rs-online.com		
25	SW3-1	Potentiometer 10k ohm	LCD back light ON	2.74
		Panel Mount Through Hole	Panel ID: 1	
		RS Stock No. 729-3555	Mounting hole dia. 7mm	

continued ...

... continued

Item	Part ID	Item	Remark	Cost (£)
		Mfr. Part No. RK0971114Z07.		
		Link: http://uk.rs-online.com		
		Momentary switch plus 10k ohm potentiometer		
26	SW3-2	Sifam pointer knob	Black, with white indicator	2.89
		RS Stock No. 799-9402	Dia. 21mm, shaft 6mm	
		Brand: Sifam		
		Mfr. Part No.SP211 006 Black		
		Link: http://uk.rs-online.com		
27	SW4	Stainless steel momentary switch	Reset	3.49
		Model: 208	Panel ID: 9	
		http://www.modmaker.co.uk	Mounting hole dia. 12mm	
28	SW5	Rocker switch 1-pin OFF	Data log ON/OFF	1.20
		Item No: WIPPE 1881.1103	Panel ID: 7	
		Mfr: Marquardt	Mounting holes dia: 20mm	
		Link: http://www.reichelt.com		
29	DR1	Perma-Proto Mint Tin Size Breadboard PCB		5.38
		SKU: PPADA723		
		Brand: Adafruit		
		Part No. 571		
		Link: https://www.proto-pic.co.uk		
30	BR3-15	DC Barrel Jack Adapter – Breadboard Compatible	Signal Conditioning Circuit Item	0.84
		Part No. PRT-10811		
		Brand: Sparkfun		
		Link: https://www.proto-pic.co.uk		
31	ACAC1	Mascot AC/AC Adaptor Type 9580	Input: 230VAC 50Hz	11.29
		Part No. 9580900600	Output: 9VAC 300MA	
		Link: http://uk.rs-online.com		

continued ...

... continued

Item	Part ID	Item	Remark	Cost (£)
32	ANT1	3V GPS Antenna Magnetic Mount SMA Part No. RB-Spa-135 Link: http://www.robotshop.com	Male SMA connector 5m cable	12.43
33	TB1	Screw Terminal Block 2x2way eBay	2-Pin Plug-in Terminal Block Screw PCB Connector Pitch 2.54mm Green	0.47
34	TB2	Screw Terminal Block 2x2way eBay	2-Pin Plug-in Terminal Block Screw PCB Connector Pitch 2.54mm Green	0.47
35	TB3	Screw Terminal Block 8way eBay	8-Pin Plug-in Terminal Block Screw PCB Connector Pitch 2.54mm Green	1.75
36	TB4	Screw Terminal Block 8way eBay	8-Pin Plug-in Terminal Block Screw PCB Connector Pitch 2.54mm Green	1.75
37	TB5	Screw Terminal Block 8way eBay	8-Pin Plug-in Terminal Block Screw PCB Connector Pitch 2.54mm Green	1.75
38	BB1	CTLBREAK-COBB40-01 Code: 736009 Link: https://www.rapidon3.61line.com	Cyntech CTLBREAK-COBB40-01 Breadboard Breakout for Raspberry Pi Model B+/2 Complete with breakout board plus 40-pin ribbon cable Mounted inside Project Box base	3.61
39	BB2	CTLBREAK-COBB40-01 Code: 736009 Link: https://www.rapidon3.61line.com	Cyntech CTLBREAK-COBB40-01 Breadboard Breakout for Raspberry Pi Model B+/2 Complete with breakout board plus 40-pin ribbon cable Mounted inside Project Box lid	3.61
40	ENCL	TEKO Enclosures Model: 424.18	Length 237mm Width 160mm	32.24

continued ...

... continued

Item	Part ID	Item	Remark	Cost (£)
		Link: http://www.teko.co.uk/en/products/family/AL/series/42	Height 100mm	

A.2 Visual display and controls layout

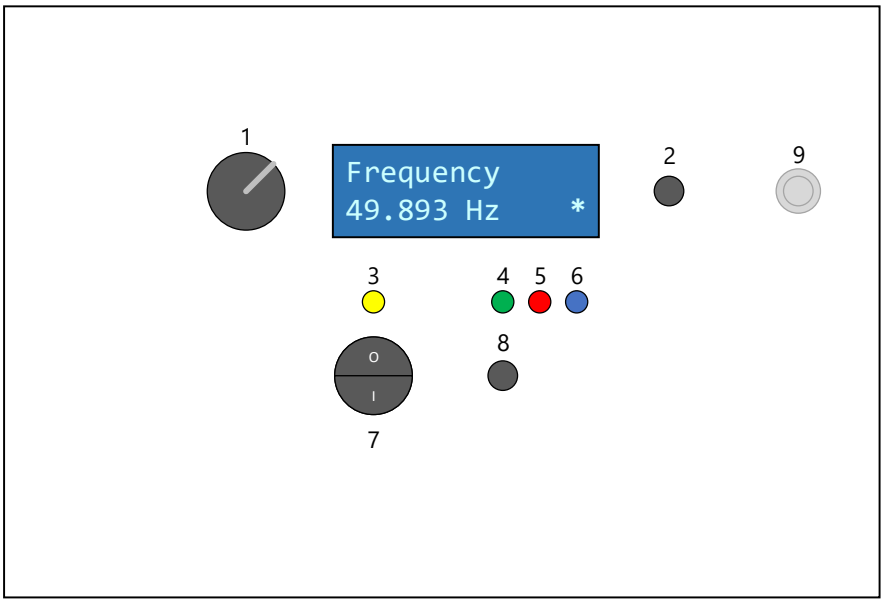


Figure A.1 Visual display and controls layout

Table A.2
Frequency measurement controls legend

Item	Type	Description
1	Linear Potentiometer (plus switch)	LCD Back Light DIM (ON)
2	Push Button Switch	Screen Cycle
3	Indicator LED Yellow	Write to on-board micro SD card
4	Indicator LED Green	Event - Operational Threshold
5	Indicator LED Red	Event - Statutory Threshold
6	Indicator LED Blue	GPS Fix
7	Rocker Switch	Data Log ON/OFF
8	Push Button Switch	Event - Reset
9	Push Button Switch	System Reset
10	LCD 16x2	Visual Display
11	GPS uFL RF Connector	Not Shown (side panel)

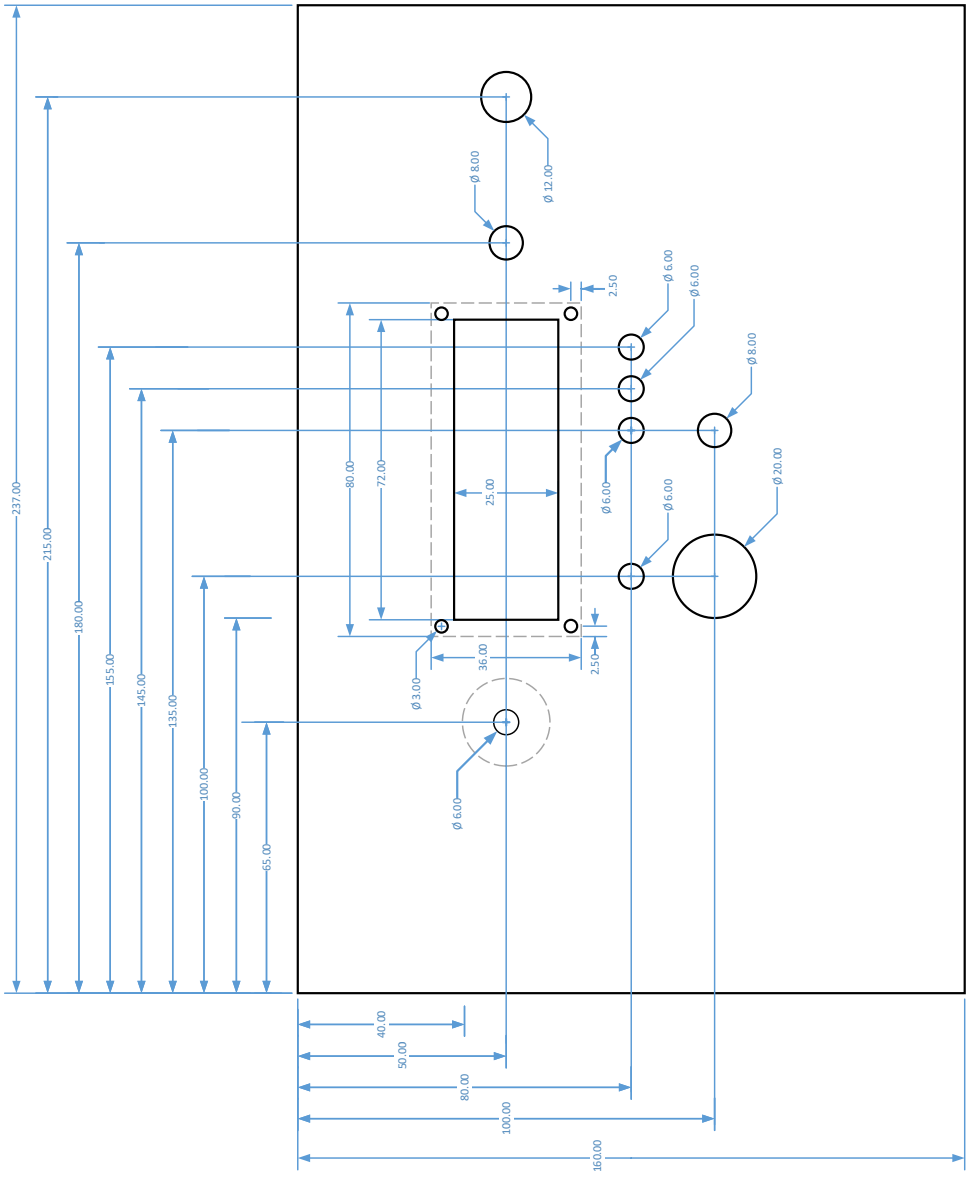


Figure A.2 Enclosure engineering drawing

A.3 PermaProto breadboard

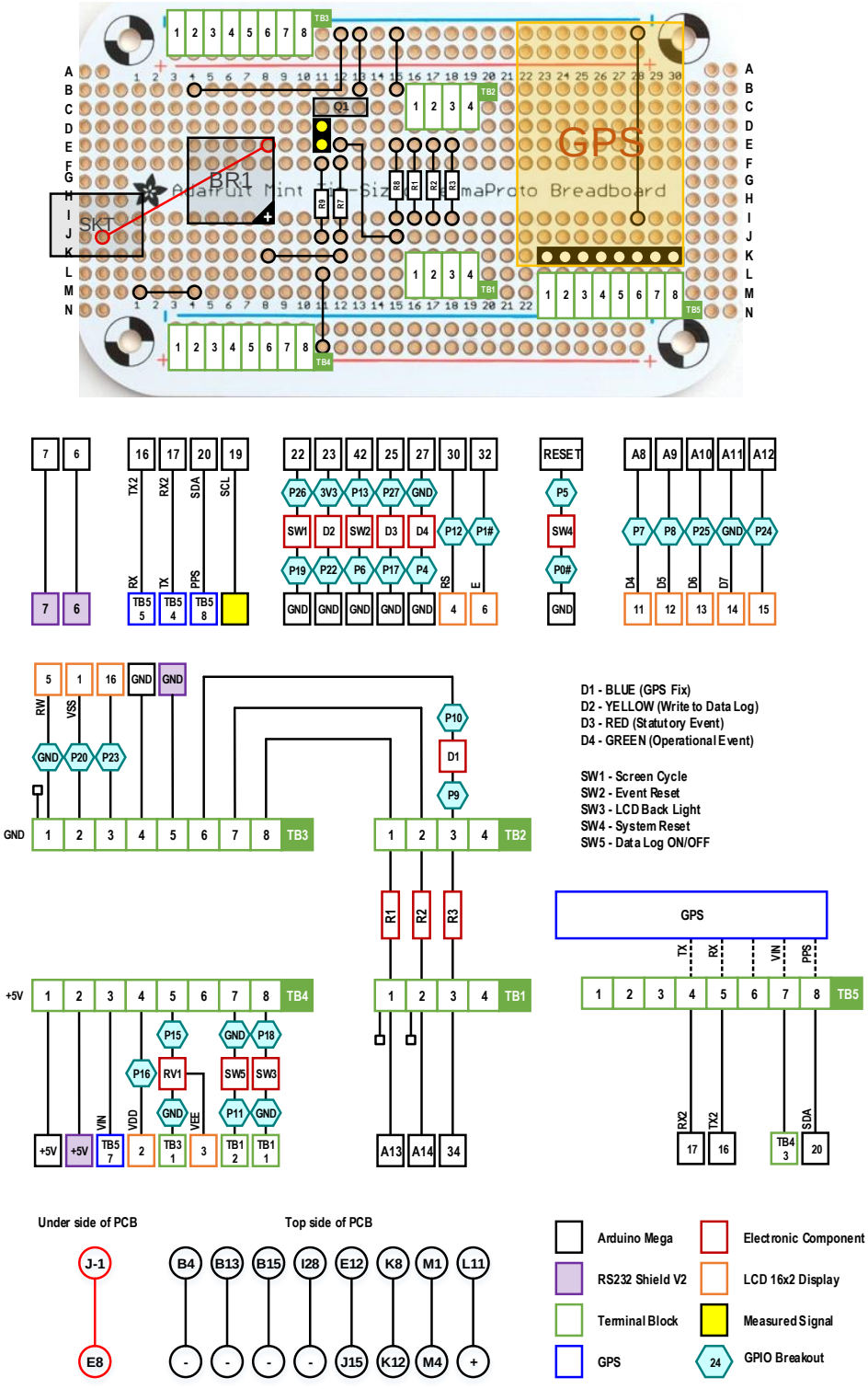


Figure A.3 PermaProto breadboard schematic

A.4 GPIO breakout board pinout

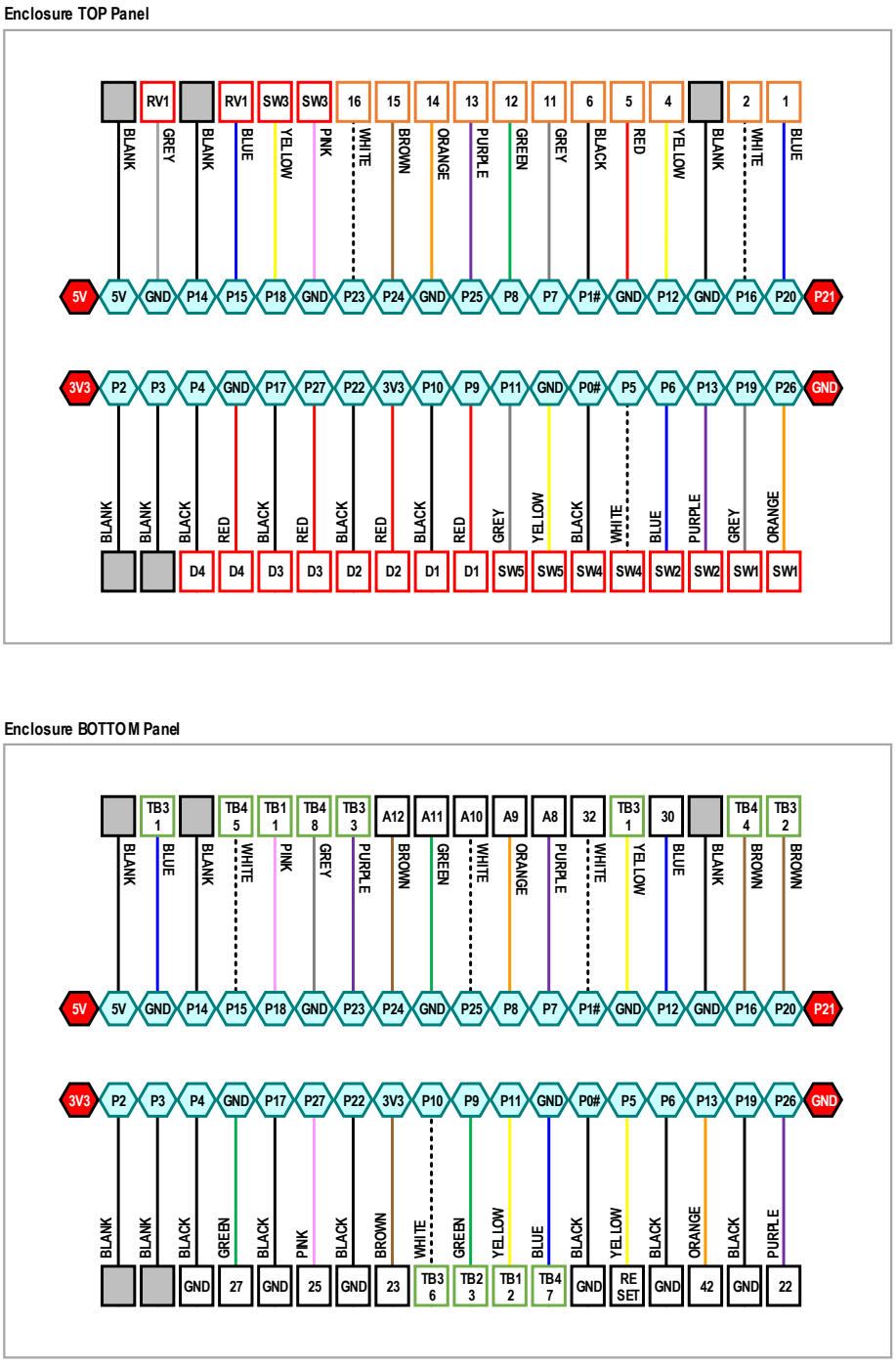


Figure A.4 GPIO breakout board pinout

A.5 Wiring schematic

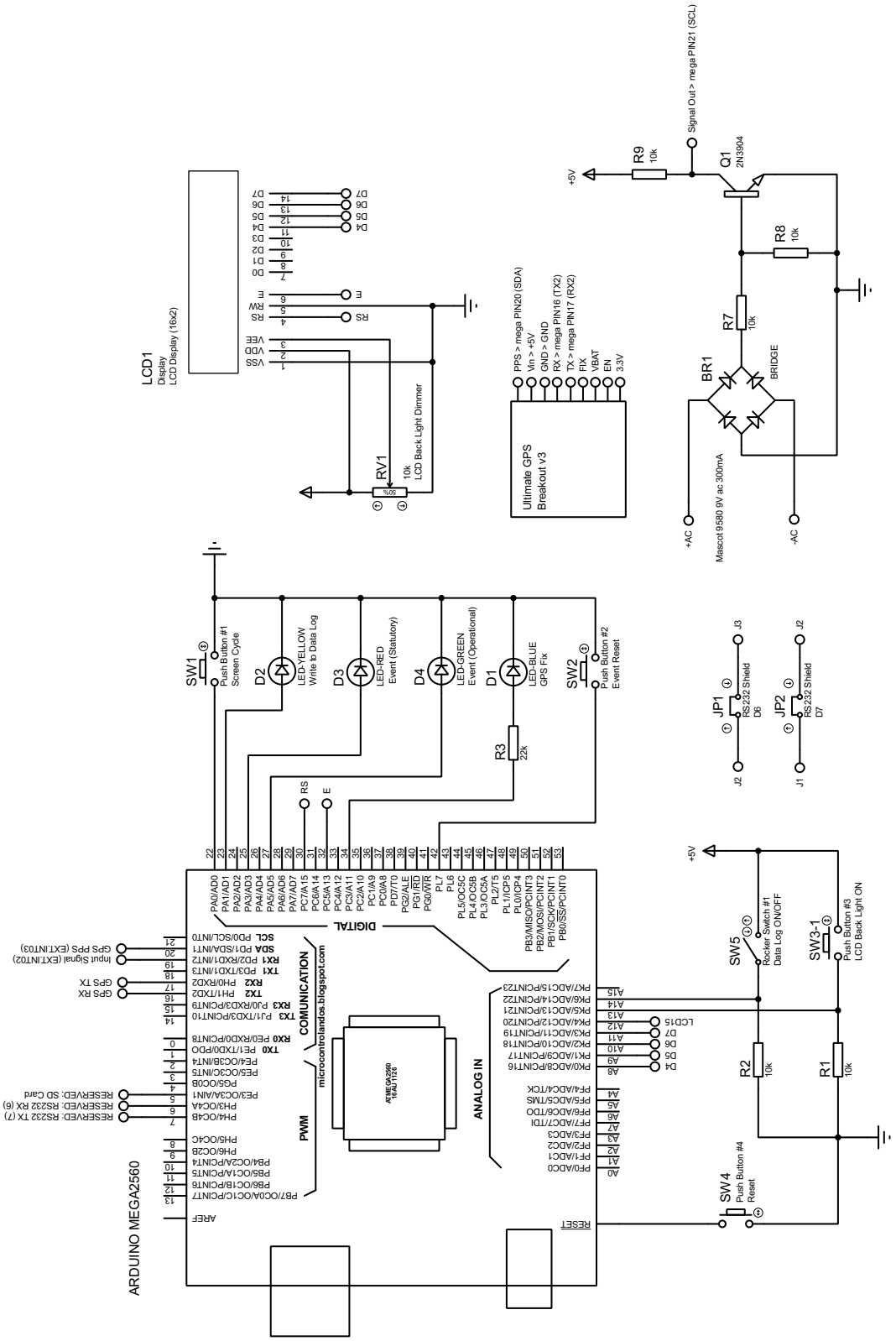
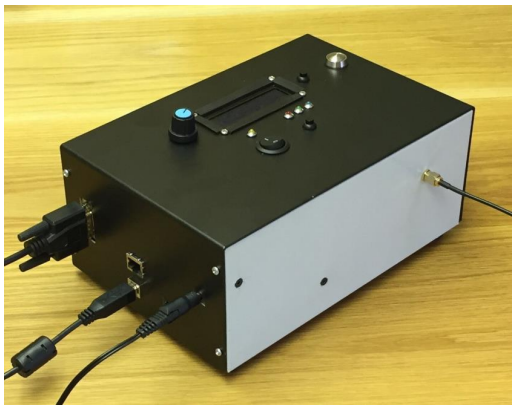
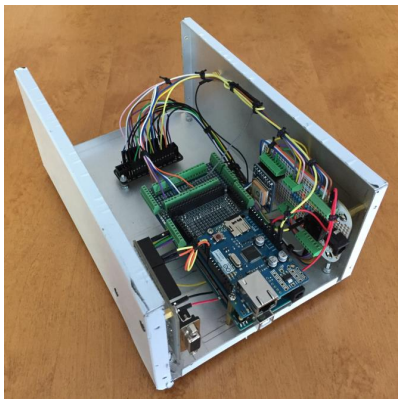


Figure A.5 Frequency measurement wiring schematic

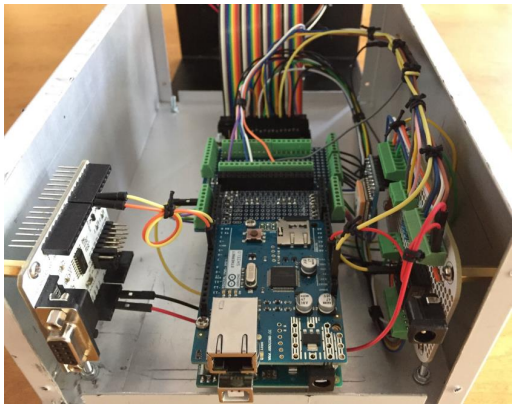
A.6 Assembly



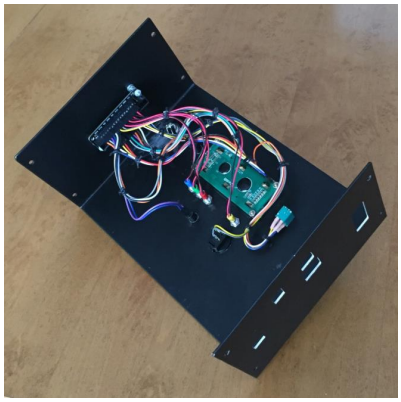
(a) External



(b) Internal (base)



(c) Internal (base)



(d) Internal (Top)

Figure A.6 Frequency measurement assembly

Appendix B

Frequency Measurement Software Development

Appendix Contents

B.1 A note about Arduino IDE

The Arduino integrated development environment (IDE) is an open-source programming tool for writing software code (referred to by Arduino users as a sketch) and uploading it to one of the Arduino family of microcontrollers. This study offers a sketch for a frequency measurement instrument written using Arduino IDE version 1.8.8. When uploaded to a low-cost Arduino Mega2560 microcontroller, this, together with several compatible external electronic components (shields), forms part of the frequency measurement instrument architecture. Later, an industrial level input/output fully-featured compatible Arduino board is introduced. Taking advantage of industrial level output channels of 0 to 10 VDC and 4 to 20 mA, the product offers a low-cost interfacing solution to bridge the gap between Arduino compatibility and external sensors and actuators during hardware-in-the-loop testing.

B.2 Arduino IDE sketch flow diagram

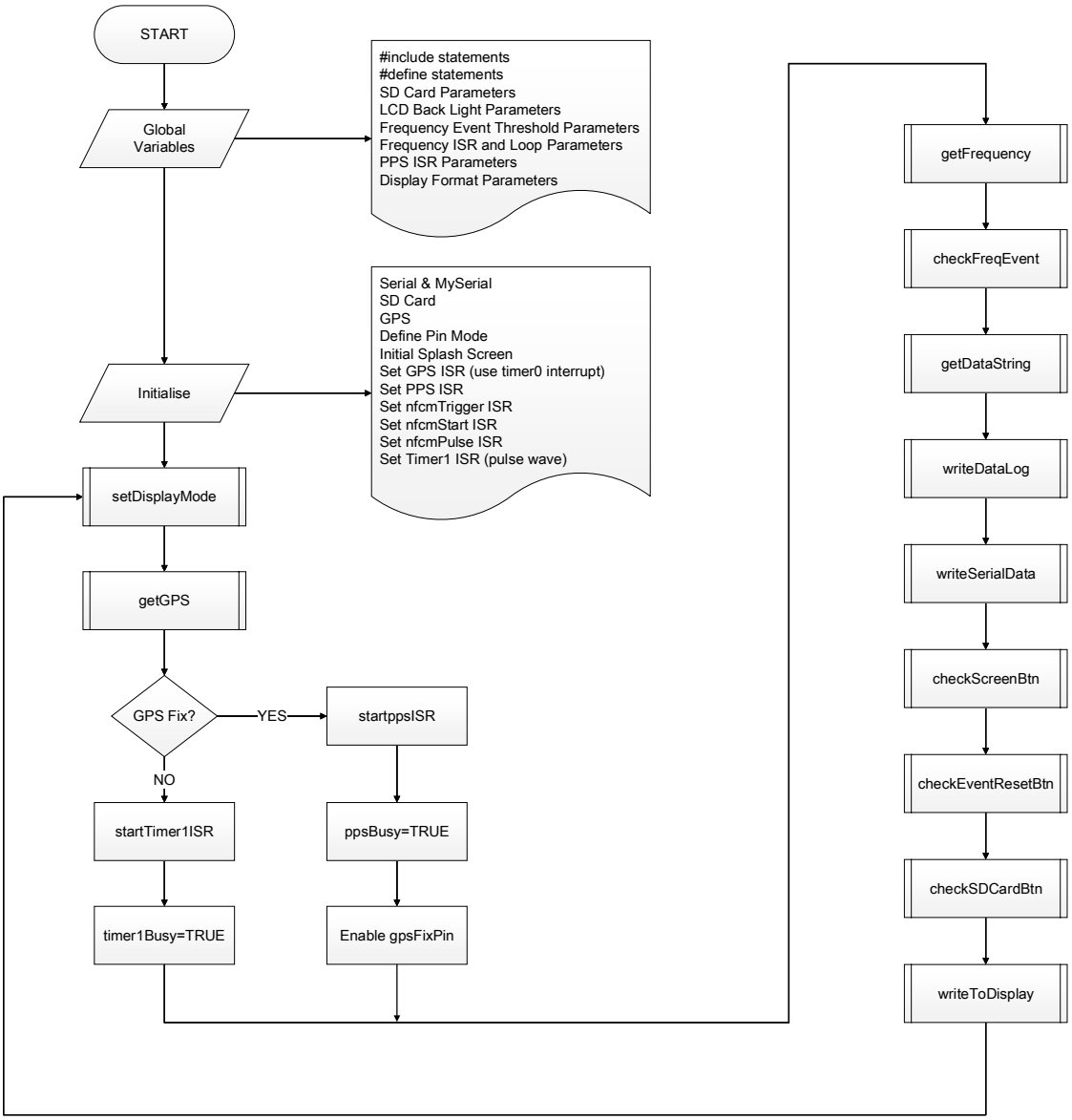


Figure B.1 Arduino IDE sketch flow diagram

B.3 Functions

Table B.1
Frequency measurement functions

Fn(n)	Function Name	Definition	Description
Fn(1)	setDisplayMode()	lcdLightOnTime lcdButtonStatus lcdBackLightON currentlcdBackLigtOnStartTime lcdBackLightPin lcdBackLightOnStart	Function will monitor status of lcdButtonStatus - digitalRead(buttonLcdPin) - and will turn on LCD back light for set period of time defined by the parameter lcdLightOnTime (*default LCD on time set to 60 seconds*) when LCD button is pressed.
Fn(2)	getGPS()	GPSECHO	Function to get GPS data. GPS interrupt is called once a millisecond, looks for any new GPS data, and then stores it. Software code provides option to change GPS lat/lon format displayed on LCD.
Fn(3)	getFrequency()	nfcmFreq nfcmCount nfcmEndTime nfcmStartTime cf freqFilter alpha	Function to calculate frequency (nfcmFreq) based on calculated time interval of pulse count. A correction factor (calibration) and format adjustment is applied before bandpass filter applied.
Fn(4)	checkFreqEvent()	freqStatutoryLow freqStatutoryHigh freqOperationalLow freqOperationalHigh freqFilter freqStatLEDPin freqOpLEDPin	Function is to check frequency measurement and if found to exceed set thresholds then provide visual indication and increment appropriate event log. Thresholds defined as Statutory (50 ±0.5 Hz) or Operational (50 ± 0.2 Hz). System will reset event log counter is count exceeds 99. • Operational threshold LED (GREEN) • Statutory threshold LED (RED)

continued ...

. . . continued

Fn(n)	Function Name	Definition	Description
		lowFreqCount highFreqCount	
Fn(5)	getDataString()	dataStringA stringYear stringMonth stringDay dataStringB stringHour stringMinute stringSecond stringMilliSeconds dataStringC stringFreq	Function to set data string output into correct format defined by: • dataStringA: yyyy-mm-ddT • dataStringB: hh:mm:ssZ • dataStringC: ff.ff For example: `2017-01-31T21:47:58Z50.12`
Fn(6)	writeDataLog()	writeToLogEnable writeToLog writeToLogInterval logLEDPin dataStringA dataStringB dataStringC	Function will write data string to on-board micro SD card. Option to pause write function provided (user switch control) to enable the micro SD card to be removed safety from the microcontroller. A carriage return is inserted at the end of each data log entry. Option to change the time interval of each write to on-board micro SD card is provided (software variable: `writeToLogInterval`). Default value is set at 1 second. Option to change the log file name is provided (software code). Default file name is `FREQLOG.txt`. A data log LED will flash every time a data string is written to the on-board micro SD card.

continued . . .

. . . continued

Fn(n)	Function Name	Definition	Description
Fn(7)	writeSerialData()	dataStringA dataStringB dataStringC	Function to write to external equipment via RS232 cable. Output to Serial Monitor also enabled to enable data transfer from frequency measurement tool to MATLAB using USB to external computer Serial connection. A visual indication (*) is displayed on the LCD display every time a data package is written to external equipment. Format of each data package is consistent as described above.
Fn(8)	checkScreenBtn()	buttonScreenPin screen	Function will change LCD screen display when Screen Cycle push button is pressed. If GPS fix is not enabled then Screen #4 will not be displayed.
Fn(9)	checkEventResetBtn()	buttonResetEventPin lowFreqCount highFreqCount freqOpLEDPin freqStatLEDPin	Function designed to allow user to reset event log and counters.
Fn(10)	checkSDCardBtn()	writeToLogEnable	Function to check the status of the write to micro SD card button. This will enable/disable write to on-board micro SD card depending status of write to micro SD card button.
Fn(11)	writeToDisplay()	screen freqFilter print3digit() lowFreqCount highFreqCount dataStringB dataStringA locationFormat GPS.latitudeDegrees GPS.lat GPSlongitudeDegrees	Function sets format and content of each LCD display screen. There are 5 screens that the user can select (cyclic): • Screen #1: Display Frequency and indicate when write to Serial Data RS232 event occurs (*). • Screen #2: Display number of Low and High Statutory events. • Screen #3: Display date and time in UTC format. • Screen #4: Display location plus altitude (If GPS fix enabled). • Screen #5: Read SD card and report back free capacity.

continued . . .

...continued

Fn(n)	Function Name	Definition	Description
		GPS.lon	
		GPS.altitude	
		writeToLogEnable	

B.4 Arduino sketch: frequency measurement tool

```
/*
Title:        Frequency Measurement Tool
Filename:     freq_meas_tool_R5.ino
Prepared by:  Sean Williams
Modified:     11 October 2017
Description:  Designed to measure UK mains electrical frequency at 1 mHz or 10 mHz resolution.

              Main features include:
              1. Write to data log (micro SD Card) - fixed format at user defined interval
              2. Write to external device using RS232 cable - fixed format at 1 Hz
              3. Display GPS location (lat/lon/alt) when fix acquired
              4. Monitor and record frequency events
              5. Power saving features including lcd back light auto time out
              6. Turn ON/OFF write to micro SD Card
              7. View data log file size (only when write to SD Card selected to OFF)
              8. Output display refreshed at 1 second intervals.

----- LED indicators -----
Yellow - Pulse when write to on-board data log (micro SD Card).
Green - Latch when frequency exceeds low (49.8 Hz) or high (50.2 Hz) operational thresholds.
Red - Latch when frequency exceeds low (49.5 Hz) or high (50.5 Hz) statutory thresholds.
Blue - Pulse 1 per 15 seconds when gps fix acquired.

----- buttons -----
:Screen Cycle > 22 - Momentary push button when depressed will cycle LCD screens (0 to 5)
:Event Reset > 42 - Momentary push button when depressed will extinguish Red and Green LED
 and reset LOW and HIGH counters (screen#1)
:LCD Backlight Time On > A13 - Momentary push button when depressed will illuminated LED
 back light and display for set time period (default 60 seconds)
:Arduino mega 2560 soft reset > RESET - Momentary push button when depressed will instruct
 Arduino IDE to restart the AVR
:Write to micro SD Card > A14 - Rocker switch to turn ON/OFF write to on-board data
 log (micro SD Card).

----- pin out -----
GPS RX > TX2(16) Serial2
GPS TX > RX2(17) Serial2
GPS PPS > EXT.INT03 (SDA 20)
INPUT SIGNAL > EXT.INT04 (RX1 19)
SD Card > (5) reserved
RS232 > (6) reserved
RS232 > (7) reserved

----- lcd pin out ------
1  |  2  |   3   | 4  |  5  | 6  | 11 | 12 | 13  | 14  | 15  | 16
VSS | VDD | VO    | RS | RW  | E  | D4 | D5 | D6  | D7  | K   | Pot out
GND | +5V | Potin | 30 | GND | 32 | A8 | A9 | A10 | A11 | A12 | GND
*/

String version = "freq_meas_tool_R5"; // write version number to Serial Monitor on startup

#include <SPI.h>
#include <Wire.h>
#include <SD.h>
#include <Adafruit_GPS.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>

#define myGPSSerial Serial2 // define hardware connection GPS TX->RX2(17) & GPS RX->TX2(16)
#define GPSECHO false // set false to turn off extra GPS data display
#define buttonScreenPin 22 //set pin 22 for screen cycle push button
#define buttonResetEventPin 42 // set pin 42 for reset event push button
#define logLEDPin 23 // set pin 23 for data log LED
#define freqStatLEDPin 25 // set pin 25 for statutory frequency event RED LED
#define freqOpLEDPin 27 // set pin 27 for operational frequency event GREEN LED
#define debounce 50 // button debounce
#define nfcmSample 10 // defines the number of pulse counts in new freq calc method (nfcm)
    //calculation
#define lcdBackLightPin A12 // set pin A12 for lcd back light
#define buttonLcdPin A13 // set pin A13 for timed on lcd back light
#define writeToLogPin A14 // set pin A14 for write to micro SD Card ON/OFF rocker switch
#define gpsFixPin 34 // set pin 34 for gps fix LED
#define gpsFixInterval 15 // set time interval between each gps fix LED flash (seconds)

LiquidCrystal lcd(30, 32, A8, A9, A10, A11); // set lcd pin out
SoftwareSerial mySerial (6, 7); // set pins 6 and 7 for RS232 Serial TX/RX **DO NOT CHANGE**
```

```
79  // RS232 Shield V2 jumper setting D6(J2J3), D7(J1J2)
80  Adafruit_GPS GPS(&myGPSSerial);
81
82  // ----- SD card parameters -----
83  int writeToLog = 0; // check parameter equals time interval before write to log
84  const int mySDCard = 4; // reserved for SD card **DO NOT CHANGE**
85  const int writeToLogInterval = 15; // number of seconds between each write to log (SD card) event
86  boolean writeToLogEnable; // parameter determines whether to write data to micro SD card or not
87  float bytes; // data log file size parameter
88  float kilobytes; // data log file size parameter
89  float megabytes; // data log file size parameter
90  float gigabytes; // data log file size parameter
91
92  // ----- lcd back light parameters -----
93  int lcdLightOnTime = 60; // set time lcd back light is on after button press (seconds)
94  unsigned int currentlcdBackLightOnStartTime = 0;
95  unsigned long lcdBackLightOnStart;
96  boolean lcdBackLightOn;
97  int lcdButtonStatus = 0;
98
99  // ----- frequency event threshold parameters -----
100 const double freqStatutoryLow = 49.5; // set statutory low frequency event threshold
101 const double freqStatutoryHigh = 50.5; // set statutory high frequency event threshold
102 const double freqOperationalLow = 49.8; // set operational low frequency event threshold
103 const double freqOperationalHigh = 50.2; // set operational high frequency event threshold
104
105 // ----- frequency isr and loop parameters -----
106 volatile int nfcmCount; // new freq calc method (nfcm) counter
107 volatile unsigned long nfcmStartTime; // new freq calc method (nfcm) start time
108 volatile unsigned long nfcmEndTime; // new freq calc method (nfcm) end time
109 volatile boolean startTimer1ISR = true;
110 volatile boolean timer1Busy = true;
111 volatile boolean rocofFlag = false; // use rocof algorithm only when true
112 volatile int rocofCount = 0;
113 const int rocofThreshold = 65; // rate of change of frequency threshold
114 float nfcmFreq = 0; // new freq calc method (nfcm) frequency
115 float freqFilter = 50e3;
116 const double alpha = 0.7258; // frequency filter
117 const double cf = 0.999646; // frequency correction factor
118 unsigned int Hz = 0;
119 unsigned int mHz = 0;
120 int lowFreqCount = 0; // low frequency event counter
121 int highFreqCount = 0; // high frequency event counter
122
123 // ----- pps isr parameters -----
124 volatile boolean ppsStart = false;
125 volatile boolean startppsISR = true;
126 volatile boolean ppsBusy = true;
127 boolean timer0Busy = false;
128 void useTimer0Interrupt(boolean);
129 const byte ppsPin = 20;
130 const byte inputPin = 19;
131 int ppsStartCount = 0;
132
133 // ----- display format parameters -----
134 boolean toggle1 = true; // write to RS232 indicator on Screen #0 toggle1
135 int toggle1Count = 0;
136 boolean toggle2 = false; // gps fix indicator, flash LED once every 15 seconds
137 int toggle2Count = 0;
138
139 boolean locationFormat = true; // set default location format to display:
140 // true = DD (decimal degrees) googlemaps format
141 // false = Lat: DDMM.MMMM Long: DDDMM.MMMM
142 int screen = 0; // default initial screen to display:
143 // Screen #0  Frequency
144 //            ff.ff[f] Hz   *   (option 10mHz or 1mHz)
145 //                              (* = flash when write to serial port RS232)
146 // Screen #1  Low: nn
147 //            High: nn
148 // Screen #2  T: hh:mm:ss  UTC
149 //            D: yyyy-mm-dd
150 // Screen #3a Lat: nn.nnnnN     if (gps.fix)
151 //            Long: n.nnnnW
152 // Screen #3b No GPS Fix        if (!gps.fix) miss out Screen #4
153 //            [blank line]
154 // Screen #4  Alt: nnnn.nn      if (gps.fix)
155 //            [blank line]
156 // Screen #5a SD File Size      if (SD Card Switch ON)
157 //            Turn off to read
158 // Screen #5b SD File Size      if (SD Card Switch OFF)
159 //            nnn.nn [GB] [MB] [KB] [Bytes]
160
161 int b1, b2; // button debounce
162 String dataStringA, dataStringB, dataStringC; // place holders for data string
163 // dataStringA = <yyyy-mm-ddT
164 // dataStringB = hh:mm:ssZ,
```

```
// dataStringC = ff.ff>
String stringYear, stringMonth, stringDay; //dataStringA
String stringHour, stringMinute, stringSeconds, stringMilliSeconds; //dataStringB
String stringFreq; //dataStringC

void setup() {
  // ----- initialize Serial & mySerial -----
  Serial.begin(9600);
  mySerial.begin(9600);
  while (!Serial) {
    ; // wait for Serial port to connect.
  }
  while (!mySerial) {
    ; // wait for mySerial port to connect.
  }

  // ----- initialize SD Card -----
  //Serial.println(version);
  if (!SD.begin(mySDCard)) {
    return;
  }

  // ----- initialize GPS -----
  GPS.begin(9600);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA); // set National Marine Electronics
      // Association sentence
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_5HZ); // possible to change refesh rate in
      // header file
  GPS.sendCommand(PGCMD_ANTENNA);
  useTimer0Interrupt(true);
  myGPSSerial.println(PMTK_Q_RELEASE);

  // ----- define pin mode -----
  pinMode(buttonScreenPin, INPUT); // screen cycle
  digitalWrite(buttonScreenPin, HIGH);
  pinMode(buttonResetEventPin, INPUT); // reset event counters
  digitalWrite(buttonResetEventPin, HIGH);
  pinMode(logLEDPin, OUTPUT); // data log write LED
  pinMode(freqOpLEDPin, OUTPUT); // frequency event flag LED
  pinMode(freqStatLEDPin, OUTPUT); // exceed frequency threshold LED
  pinMode(ppsPin, INPUT); // pps input
  pinMode(inputPin, INPUT); // input signal (frequency)
  pinMode(buttonLcdPin, INPUT); // turn on lcd back light
  pinMode(lcdBackLightPin, OUTPUT); // lcd back light
  digitalWrite (lcdBackLightPin, HIGH);
  lcdBackLightOn = true; // turn on lcd back light
  pinMode(writeToLogPin, INPUT); // write to log
  writeToLogEnable = true; // enable write to log (micro SD card)
  pinMode(gpsFixPin, OUTPUT);

  // ----- set timer1 interrupt at 1Hz -----
  cli(); // stop interrupts
  TCCR1A = 0; // set entire TCCR1A register to 0
  TCCR1B = 0; // same for TCCR1B
  TCNT1  = 0; //initialize counter value to 0
  OCR1A = 15624; // set compare match register for 1 Hz increments = (16*10^6)/(1*1024)-1
      // (must be <65536)
  TCCR1B |= (1 << WGM12); // turn on CTC mode
  TCCR1B |= (1 << CS12) | (1 << CS10); // Set CS12 and CS10 bits for 1024 prescaler
  TIMSK1 |= (1 << OCIE1A); // enable timer compare interrupt
  sei();//allow interrupts

  // ----- initial splash screen -----
  lcd.begin(16, 2);
  lcd.setCursor(3, 0);
  lcd.print("DTA Energy");
  lcd.setCursor(2, 1);
  lcd.print("Teesside Uni");
  delay(1500);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Freq Meas Tool");
  lcd.setCursor(0, 2); lcd.print("Version: ");
  lcd.print(String(version).substring(15, 17));
  delay(3000);
  lcd.clear();

  // LED check
  digitalWrite(logLEDPin, HIGH);
  digitalWrite(freqOpLEDPin, HIGH);
  digitalWrite(freqStatLEDPin, HIGH);
  digitalWrite(gpsFixPin, HIGH);
  delay (2000);
  digitalWrite(logLEDPin, LOW);
```

```
249     digitalWrite(freqOpLEDPin, LOW);
250     digitalWrite(freqStatLEDPin, LOW);
251     digitalWrite(gpsFixPin, LOW);
252     delay (2000);
253
254     attachInterrupt(digitalPinToInterrupt(inputPin), nfcmTrigger, FALLING); // start isr
255   }
256
257   void loop() {
258     setDisplayMode(); // fn to control lcd back light
259     getGPS(); // fn to get gps data
260     // if gps fix then use pps as time stamp to carry out the following, else use timer1
261     // set at 1 second intervals as time stamp:
262     //      1. get frequency
263     //      2. check frequency against set threshold limits
264     //      3. set data into correct format in advance to data logging and RS232 serial output
265     //      4. write data to on-board micro SD card
266     //      5. write data to external via RS232
267     if (GPS.fix) {
268       if (startppsISR) attachInterrupt(digitalPinToInterrupt(ppsPin), pps, FALLING);
269       if (!ppsBusy) {
270         if (nfcmCount == nfcmSample) {
271           if (toggle2Count == gpsFixInterval) { // set gps fix LED ON if at set time interval
272             digitalWrite(gpsFixPin, HIGH);
273             toggle2Count = 0;
274           }
275           toggle2Count++;
276
277           getFrequency(); // fn to get calculate frequency
278           checkFreqEvent(); // fn to check frequency against event thresholds
279           getDataString(); // fn to format date time group and frequency
280           writeDataLogger(); // fn to write data to SD card
281           writeSerialData(); // fn to write data to RS232 serial output
282           attachInterrupt(digitalPinToInterrupt(inputPin), nfcmTrigger, FALLING); // start isr
283           ppsBusy = true; // set ppsBusy to prevent if-else loop running until next pps interrupt
284           digitalWrite(gpsFixPin, LOW);
285         }
286       }
287     }
288     else {
289       // start isr
290       if (startTimer1ISR) attachInterrupt(digitalPinToInterrupt(inputPin), nfcmTrigger, FALLING);
291       if (!timer1Busy) {
292         if (nfcmCount == nfcmSample) {
293           getFrequency(); // fn to get calculate frequency (correction factor and filter applied)
294           checkFreqEvent(); // fn to check frequency against event thresholds
295           getDataString(); // fn to format date time group and frequency
296           writeDataLogger(); // fn to write data to SD card
297           writeSerialData(); // fn to write data to RS232 serial output
298           attachInterrupt(digitalPinToInterrupt(inputPin), nfcmTrigger, FALLING); // start isr
299         }
300         timer1Busy = true; // set timer1Busy to prevent if-else loop running until
301             // next timer1 interrupt
302       }
303     }
304     checkScreenBtn(); // check status of screen cycle button
305     checkEventResetBtn(); // check status of event reset button
306     checkSDCardBtn(); // check status of sd card on/off button
307     writeToDisplay(); // write data to the lcd display
308   }
309
310   void setDisplayMode() { // fn to turn on lcd back light for set period of time (lcdLightOnTime)
311     lcdButtonStatus = digitalRead(buttonLcdPin);
312     if (lcdButtonStatus == HIGH) {
313       lcdBackLightOnStart = millis();
314       currentlcdBackLightOnStartTime = 0;
315       lcdBackLightOn = true;
316       digitalWrite (lcdBackLightPin, HIGH);
317       lcd.display();
318     }
319     else {
320       if (lcdBackLightOn) {
321         currentlcdBackLightOnStartTime = millis() - lcdBackLightOnStart;
322         if (currentlcdBackLightOnStartTime > (lcdLightOnTime * 1e3)) {
323           lcdBackLightOn = false;
324           digitalWrite(lcdBackLightPin, LOW);
325           lcd.noDisplay();
326         }
327       }
328     }
329   }
330
331   void getGPS() {  // fn to get GPS data
332     if (!timer0Busy) {
333       // read data from the GPS in the 'main loop'
```

```
334       char c = GPS.read();
335       // if you want to debug, this is a good time to do it!
336       if (GPSECHO)
337         if (c) Serial.print(c);
338     }
339     // if a sentence is received, we can check the checksum, parse it...
340     if (GPS.newNMEAreceived()) {
341       if (!GPS.parse(GPS.lastNMEA()))  // this sets the newNMEAreceived() flag to false
342         return;
343     }
344   }
345
346   void getFrequency () { // fn to calculate frequency
347     nfcmFreq = (1e6 * nfcmCount) / (2 * (nfcmEndTime - nfcmStartTime)); // calculate frequency
348     nfcmCount = 0; // reset the input signal pulse counter
349     nfcmFreq = nfcmFreq * cf * 1e3; // apply correction factor and change format (1e3)
350     // limit value of new frequency to previous value if rate of change exceeds rocofThreshold
351     // start rocof algorithm only after 4th recorded frequency after power on
352     if (rocofFlag) {
353       if (abs(freqFilter - nfcmFreq) > rocofThreshold) { // rate of change threshold
354         nfcmFreq = freqFilter; // if rocof greater than rocofThreshold set frequency
355             // to previous value
356       }
357     }
358     freqFilter = (alpha * freqFilter) + ((1 - alpha) * (nfcmFreq)); // apply filter
359     // set flag to start rocof algorithm after 4th recorded frequency after power on.
360     if (rocofCount < 4) rocofCount++;
361     if (rocofCount > 3) rocofFlag = true; // set flag to start rocof algorithm
362   }
363
364   void checkFreqEvent() { // fn to turn on led if frequency exceeds set threshold values
365     if (((freqFilter / 1e3) ≤ freqStatutoryLow - 0.4) ||
366               ((freqFilter / 1e3) ≥ freqStatutoryHigh + 0.4)) {
367       freqFilter = 50e3; // if threshold exceeded set paremeter to 50e3
368       digitalWrite(freqStatLEDPin, HIGH); // set Statutory threshold LED (RED) ON
369     }
370     else if ((freqFilter / 1e3) < freqOperationalLow) { // detect low Operational threshold
371           // frequency event
372       lowFreqCount++; // increment Operational threshold counter
373       if (lowFreqCount ≥ 99) lowFreqCount = 0;
374       digitalWrite(freqOpLEDPin, HIGH); // set Operational threshold LED (GREEN) ON
375       return;
376     }
377     else if ((freqFilter / 1e3) > freqOperationalHigh) { // detect high Operational threshold
378           // frequency event
379       highFreqCount++; // increment Operational threshold counter
380       if (highFreqCount ≥ 99) highFreqCount = 0;
381       digitalWrite(freqOpLEDPin, HIGH); // set Operational threshold LED (GREEN) ON
382       return;
383     }
384   }
385
386   void getDataString() { // fn to set data into correct format
387     // dataString defined by dataStringA, dataStringB and dataStringC
388
389     // dataStringA: <yyyy-mm-ddT
390     stringYear = ("20") + String(GPS.year);
391     stringMonth = GPS.month;
392     if (stringMonth.length() == 1) stringMonth = "0" + stringMonth;
393     stringDay = GPS.day;
394     if (stringDay.length() == 1) stringDay = "0" + stringDay;
395     dataStringA = "";
396     dataStringA += "<";
397     dataStringA += stringYear;
398     dataStringA += "-";
399     dataStringA += stringMonth;
400     dataStringA += "-";
401     dataStringA += stringDay;
402     dataStringA += "T";
403
404     // dataStringB: hh:mm:ssZ,
405     // ss is set in pps and timer1
406     stringHour = GPS.hour;
407     if (stringHour.length() == 1) stringHour = "0" + stringHour;
408     stringMinute = GPS.minute;
409     if (stringMinute.length() == 1) stringMinute = "0" + stringMinute;
410     dataStringB = "";
411     dataStringB += stringHour; dataStringB += ":";
412     dataStringB += stringMinute; dataStringB += ":";
413     dataStringB += stringSeconds; dataStringB += "Z,";
414
415     // dataStringC: ff.ff>
416     stringFreq = (freqFilter / 1e3);
417     dataStringC = "";
418     dataStringC += stringFreq;
419     dataStringC += ">";
420   }
```

```
421
422  void writeDataLogger() { // fn to write data string to micro SD card
423    if (writeToLogEnable == HIGH) { // only write if writeToLogEnable is true
424      if (writeToLog == writeToLogInterval) { // only write to log at set time interval set
425            // by writeToLog
426        // data string format: "yyyy-mm-ddThh:mm:ssZff.ff"
427        if (dataStringA.length() == 12) { // test dataString for correct format '<yyyy-mm-ddT',
428              // if fail dont write to log
429          digitalWrite(logLEDPin, HIGH); // set data logger led to ON
430          File dataFile = SD.open("freqlog.txt", FILE_WRITE); // file name on microSD card is
431              // "freqlog.txt"
432          // if the file is available, write to it:
433          if (dataFile) {
434            dataFile.print(dataStringA);
435            dataFile.print(dataStringB);
436            dataFile.println(dataStringC); // include carriage return at end of each data
437                // log entry
438            dataFile.close();
439            digitalWrite(logLEDPin, LOW); // set data logger led to OFF
440          }
441          // if the file isn't open, pop up an error:
442          else {
443            //Serial.println(F("error opening freqlog.txt"));
444          }
445        }
446        writeToLog = 0;
447      }
448      writeToLog++;
449    }
450  }
451
452  void writeSerialData() { // fn to write data to external equipment via RS232 cable
453    if (dataStringA.length() == 12) { // test dataString for correct format yyyy-mm-dd*
454      if (toggle1Count == 1) { // set rate of write Serial Data indicator on Screen #0
455        toggle1 = !toggle1;
456        toggle1Count = 0;
457      }
458      mySerial.print(String(dataStringA)); //write <yyyy-mm-ddT
459      mySerial.print(String(dataStringB)); //write hh:mm:ssZ,
460      mySerial.println(String(dataStringC)); //write ff.ff>
461
462      // next three lines writes to Serial Monitor, aim to read from USB into MATLAB
463      Serial.print(String(dataStringA)); //write <yyyy-mm-ddT
464      Serial.print(String(dataStringB)); //write hh:mm:ssZ,
465      Serial.println(String(dataStringC)); //write ff.ff>
466
467      toggle1Count++;
468    }
469  }
470
471  void checkScreenBtn() { // fn to change lcd screen display (16x2) when depressed:
472    if (!digitalRead(buttonScreenPin) && !b1) { // check screen cycle button status
473      lcd.clear(); // clear lcd display
474      screen++;
475      if ((!GPS.fix) && (screen == 4)) screen = 5; // do not display Screen #4 if no gps fix
476      if (screen == 6) screen = 0; // reset cycle to start at Screen #1
477      b1 = debounce;
478    }
479    if (!b1 == 0) b1--;
480  }
481
482  void checkEventResetBtn() { // fn to reset event led and counter when depressed
483    if (!digitalRead(buttonResetEventPin) && !b2) { // check event counter reset button status
484      lcd.clear();
485      lowFreqCount = 0; // reset low frequency counter to zero
486      highFreqCount = 0; // reset high frequency counter to zero
487      digitalWrite(freqOpLEDPin, LOW); // set event trigger led to OFF
488      digitalWrite(freqStatLEDPin, LOW); // set threshold led to OFF
489      b2 = debounce;
490    }
491    if (!b2 == 0) b2--;
492  }
493
494  void checkSDCardBtn() { // check status of write to micro SD Card button
495    writeToLogEnable = digitalRead(writeToLogPin); // read write to micro SD card button (ON/OFF)
496  }
497
498  void writeToDisplay() { // fn to set each screen display content
499    switch (screen) {
500      case 0: // display frequency
501        lcd.setCursor(0, 0);
502        lcd.print("Frequency       ");
503        lcd.setCursor(0, 1);
504
505        // Uncomment next 4 lines to display frequency ff.fff Hz
```

```
506         Hz = (freqFilter / 1e3);
507         mHz = freqFilter - (Hz * 1e3);
508         lcd.print(Hz); lcd.print('.');
509         print3digit(mHz, '0'); lcd.print(" Hz");
510
511         // uncomment next 1 line below to display frequency ff.ff Hz
512         //lcd.print(freqFilter/1e3); lcd.print(" Hz");
513
514         lcd.setCursor(15, 1); // write to Serial Data RS232 symbol ON/OFF at 1 Hz
515         if (toggle1) lcd.print("*");
516         if (!toggle1) lcd.print(" ");
517         break;
518
519       case 1:  // display number of Low and High statutory events
520         lcd.setCursor(0, 0);
521         lcd.print("Low: "); lcd.print(lowFreqCount);
522         lcd.setCursor(0, 1);
523         lcd.print("High: "); lcd.print(highFreqCount);
524         break;
525
526       case 2:  // display date and time in UTC (reverts to RTC if no GPS fix)
527         lcd.setCursor(0, 0);
528         lcd.print("T: "); lcd.print(dataStringB.substring(0, 8));
529         lcd.setCursor(13, 0);
530         lcd.print("UTC");
531         lcd.setCursor(0, 1);
532         lcd.print("D: "); lcd.print(dataStringA.substring(1, 11));
533         break;
534
535       case 3:  // display location
536         if (GPS.fix) { // if gps fix then display lat/lon
537           if (locationFormat) { //Location format: DD.DDDDDD (Google Maps)
538             lcd.setCursor(0, 0);
539             lcd.print("Lat: "); lcd.print(GPS.latitudeDegrees, 4); lcd.print(GPS.lat);
540             lcd.setCursor(0, 1);
541             lcd.print("Lon: "); lcd.print(GPS.longitudeDegrees, 4); lcd.print(GPS.lon);
542           }
543           else { //Location format: DDDMM.MMMM (NSWE)
544             lcd.setCursor(0, 0);
545             lcd.print(F("Lat: ")); lcd.print(GPS.latitude, 4); lcd.print(GPS.lat);
546             lcd.setCursor(0, 1);
547             lcd.print(F("Lon: ")); lcd.print(GPS.longitude, 4); lcd.print(GPS.lon);
548           }
549         }
550         else { // if no gps fix
551           lcd.setCursor(0, 0);
552           lcd.print(F("No GPS fix"));
553         }
554         break;
555
556       case 4:  // if gps fix then display altitude otherwise goto Screen #5
557         lcd.setCursor(0, 0);
558         lcd.print(F("Alt: ")); lcd.print(GPS.altitude); lcd.print(" m");
559         break;
560
561       case 5: // read SD card and report back free capacity
562         lcd.setCursor(0, 0);
563         lcd.print(F("SD File Size"));
564         lcd.setCursor(0, 1);
565         if (writeToLogEnable == HIGH) {
566           lcd.print(F("Turn OFF to read"));
567         }
568         else {
569           File dataFile = SD.open("freqlog.txt");
570           if ((dataFile.size() / 1024) < 1) {
571             bytes = dataFile.size();
572           }
573           else { // determine file size of data log file on micro SD Card
574             kilobytes = dataFile.size() / float(1024);
575             megabytes = kilobytes / float(1024);
576             gigabytes = megabytes / float(1024);
577           }
578           //          Serial.print("GB: ");Serial.println(gigabytes,DEC); // debug only
579           //          Serial.print("MB: ");Serial.println(megabytes,DEC); // debug only
580           //          Serial.print("KB: ");Serial.println(kilobytes,DEC); // debug only
581           //          Serial.print("Bytes: ");Serial.println(bytes,DEC); // debug only
582           //          Serial.println(dataFile.size());
583
584           lcd.setCursor(0, 1);
585           if (gigabytes ≥ 1) {
586             lcd.print(gigabytes);
587             lcd.print(" GB          ");
588           }
589           else if (megabytes ≥ 1) {
```

```
590             lcd.print(megabytes);
591             lcd.print(" MB          ");
592           }
593           else if (kilobytes ≥ 1) {
594             lcd.print(kilobytes);
595             lcd.print(" KB          ");
596           }
597           else {
598             lcd.print(bytes);
599             lcd.print(" Bytes       ");
600           }
601           dataFile.close();
602         }
603         break;
604     }
605   }
606
607   void print3digit(int n, char leadingchar) { // fn to display a three digit number with
608         // padding if necessary
609     if (n < 100) lcd.print(leadingchar);
610     if (n < 10) lcd.print(leadingchar);
611     lcd.print(n);
612   }
613
614   SIGNAL(TIMER0_COMPA_vect) { // isr: GPS interrupt
615     // Interrupt is called once a millisecond, looks for any new GPS data, and stores it
616     char c = GPS.read();
617     // if you want to debug, this is a good time to do it!
618
619   #ifdef UDR0
620     if (GPSECHO)
621       if (c) UDR0 = c;
622     // writing direct to UDR0 is much much faster than Serial.print
623     // but only one character can be written at a time.
624   #endif
625   }
626
627   void useTimer0Interrupt(boolean v) { // isr: GPS interrupt
628     if (v) {
629       // timer0 is already used for millis() - we'll just interrupt somewhere
630       // in the middle and call the "Compare A" function above
631       OCR0A = 0xAF;
632       TIMSK0 |= _BV(OCIE0A);
633       timer0Busy = true;
634     } else {
635       // do not call the interrupt function COMPA anymore
636       TIMSK0 &= ¬_BV(OCIE0A);
637       timer0Busy = false;
638     }
639   }
640
641   void pps() { // isr: use pps interrupt as time stamp when gps fix
642     // do this only once to set counters, avoid spurious first pulse outcomes
643     if (!ppsStart) {
644       ppsStartCount++;
645       if (ppsStartCount == 1) ppsStart = true;
646       attachInterrupt(digitalPinToInterrupt(inputPin), nfcmTrigger, FALLING);
647       return;
648     }
649     stringSeconds = GPS.seconds; // grab gps seconds for accurate time stamp
650     if (stringSeconds.length() == 1) stringSeconds = "0" + stringSeconds;
651     ppsBusy = false;
652     startppsISR = false;
653   }
654
655   void nfcmTrigger() { // isr: initial isr, enable the start isr
656     startTimer1ISR = false;
657     attachInterrupt(digitalPinToInterrupt(inputPin), nfcmStart, FALLING);
658   }
659
660   void nfcmStart() { // isr: record the start time and enable the pulse isr
661     nfcmStartTime = micros();
662     attachInterrupt(digitalPinToInterrupt(inputPin), nfcmPulse, FALLING);
663   }
664
665   void nfcmPulse() { // isr: record the end time and increment the count
666     nfcmEndTime = micros();
667     nfcmCount++;
668     if (nfcmCount == nfcmSample) detachInterrupt(digitalPinToInterrupt(inputPin));
669   }
670
671   ISR(TIMER1_COMPA_vect) { // isr: timer1 interrupt 1Hz
672     //generates pulse wave of frequency 1Hz/2 = 0.5kHz (takes two cycles for full wave)
673     stringSeconds = GPS.seconds; // grab gps seconds for accurate time stamp
674     if (stringSeconds.length() == 1) stringSeconds = "0" + stringSeconds;
```

```
675     timer1Busy = false;
676   }
```

Listing B.1 freq_meas_tool_R5.ino

B.5 HMI design

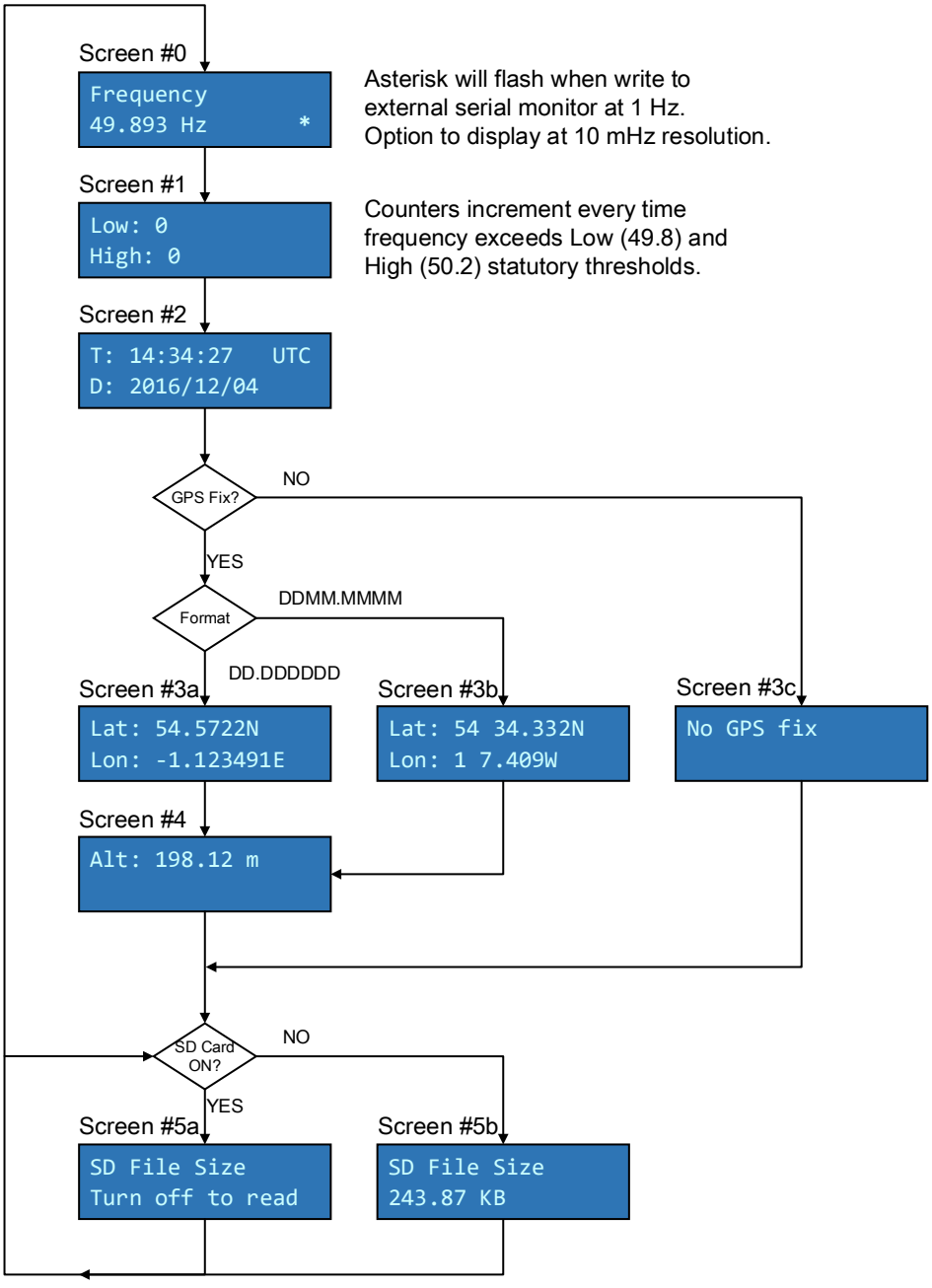


Figure B.2 Frequency measurement HMI design

B.6 Software change log

This change log contains a curated, chronologically ordered list of notable changes for each version of software designed for the Frequency Measurement Tool. Software has been developed using the Arduino IDE.

Frequency Measurement Tool R1

Filename:	freq_meas_tool_R1.ino
Arduino IDE:	1.6.12
Modified:	18-05-2017
CR Title:	N/A
Description:	Designed to measure UK mains frequency at 1 mHz or 10 mHz resolution. Main features include:

- Write to data log (micro SD Card) - fixed format at user defined interval <`yyyy-mm-dd*hh:mm:ss.sss,ff.ff#`>.
- Write to external device using RS232 cable fixed format at 1 Hz.
- Display GPS location (lat/lon/alt) when fix acquired.
- Monitor and record frequency events.
- Power saving features including LCD back light auto time out.
- Turn ON/OFF write to micro SD Card.
- View data log file size (only when write to SD Card selected to OFF).
- Output display refreshed at 1 second intervals.

Outcome: Initial formal release.

Frequency Measurement Tool R2

Filename:	freq_meas_tool_R2.ino
Arduino IDE:	1.6.12
Modified:	06-06-2017
CR Title:	CR1 - DataFormat CR2 - writeSerialData
Description:	Version freq_meas_tool_R1 writes data to the RS232 port and USB (Serial Monitor). This change amends Fn(5) `getDataString()` which defines the format of the data string output to the RS232 port. The existing format is: <`yyyy-mm-dd*hh:mm:ss.sss,ff.ff#`>. This change request requires the existing format is changed to the following new format: <`yyyy-mm-ddThh:mm:ss:ssZff.ff`>. The function Fn(5) getDataString() is defined by dataStringA, dataStringB and dataStringC, where: • dataStringA: yyyy-mm-ddT • dataStringB: hh:mm:ssZ • dataStringC: ff.ff
Outcome:	Changes to function Fn(5) `getDataString()` that define the data string output to RS232 port implemented. Visual tests carried out monitoring RS232 port using Serial Monitor, all indications found to be be correct. Updates to Frequency Measurement Plotter also carried out and tested.

Frequency Measurement Tool R3

Filename:	freq_meas_tool_R3.ino
Arduino IDE:	1.6.12
Modified:	22-06-2017
CR Title:	CR3 - ROCOF
Description:	Mains frequency is calculated at 1 second intervals using the Arduino mega 2560 internal clock (or GPS PPS if GPS fix), by sampling 10 (user defined) pulses input on an external interrupt pin (Arduino mega 2560). A passband filter is applied. A function `checkFreqEvent()` is then called, checking filtered mains frequency value. If frequency value exceeds the National Grid defined statutory threshold ± 1, the filtered mains frequency value is set to 50 and a red LED will illuminate on the control panel. In addition, if the filtered mains frequency value exceeds the National Grid operational threshold ($50.25 < F_{op} < 49.8$) a green LED will illuminate on the control panel and either a HIGH or LOW counter will increment by one (this is displayed on screen#1 on the LCD display - Figure B.2). Both red and green LED can be extinguished and counters reset to 0 only when the Event Reset button located immediately below the green LED is depressed. The LCD display screen#0 has been set to display frequency at 100 mHz resolution whereas the USB, Serial Port and data log (on-board SD Card) output at resolution of 10 mHz. During periods when the Frequency Measurement Tool has been monitoring mains frequency the red LED has been seen to illuminate, indicating the recorded frequency value has exceed the thresholds described above. Further testing revealed the Frequency Measurement Tool recorded six instances (Figure B.3) when the frequency value exceeded 51.40 during a period 2017-06-20 12:28:51 to 2017-06-22 07:15:54. A review of function Fn(4) `checkFreqEvent()` assessing whether to decrease the threshold values at which point measured frequency would be set to 50 concluded this approach would not resolve the underlying issue. An alternative approach was therefore taken; introduction of a Rate of Change of Frequency (ROCOF) algorithm. Further analysis confirmed 6 occasions where the value exceeded 51.50 exhibited a ROCOF in excess of 1.4 whereas the ROCOF for the remaining time period (153597 data entries) was ≤ 0.02 (Figure B.4). Function Fn(3) `getFrequency` was modified to monitor the calculated frequency rate of change. The same function also includes a correction factor (cf) and bandpass filter. Additional code was introduced to ensure the ROCOF algorithm started after the 4th frequency measurement was recorded; this would allow sufficient time for the Frequency Measurement Tool to stablise.

Outcome: Changes to function Fn(3) `getFrequency()` implemented.
Initial tests demonstrated the measured frequency output is stable and the sample of data analysed does not include any readings that would have resulted in the red LED illuminating. A visual test comparing sample data captured using the Frequency Measurement Tool against a sample extracted from the BMRS reporting facility for the same period of time was carried out. Instances where a ROCOF > 0.065 was observed the frequency data remained unchanged until the calculate ROCOF frequency data was < 0.065. A total of four occurrences when the calculated frequency remained constant (indicating ROCOF limits had been reached) was observed during the 3 d 18 h 46 min 32 s test period commencing 2017-06-22 13:43:30:

- 2017-06-23 05:28:05 1 h
- 2017-06-25 14:52:10 2 h 50 min
- 2017-06-26 00:00:34 54 min
- 2017-06-26 01:30:43 6 min

A 24-hour time period 2017-06-23 07:45:00 to 2017-06-24 07:45:00 where the calculated ROCOF was within limits is illustrated at Figure B.5.
Recommend to continue monitoring recorded output on visual display and set write to on-board micro SD Card switch to ON; providing option to review recorded frequency data at a later date.

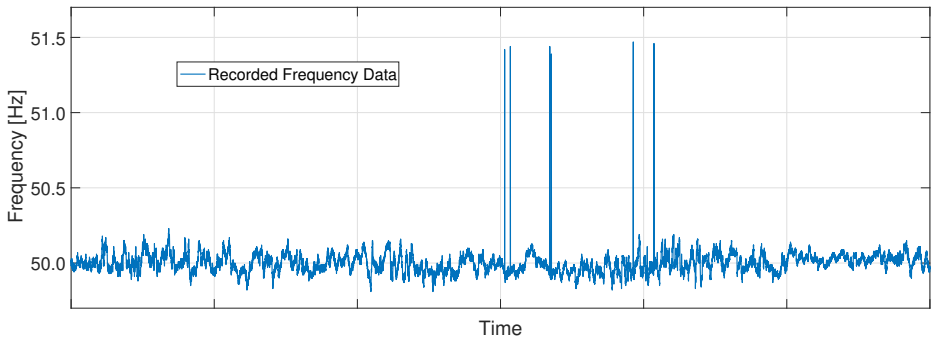


Figure B.3 Recorded frequency data
(2017-06-20 12:28:51 to 2017-06-22 07:15:54)

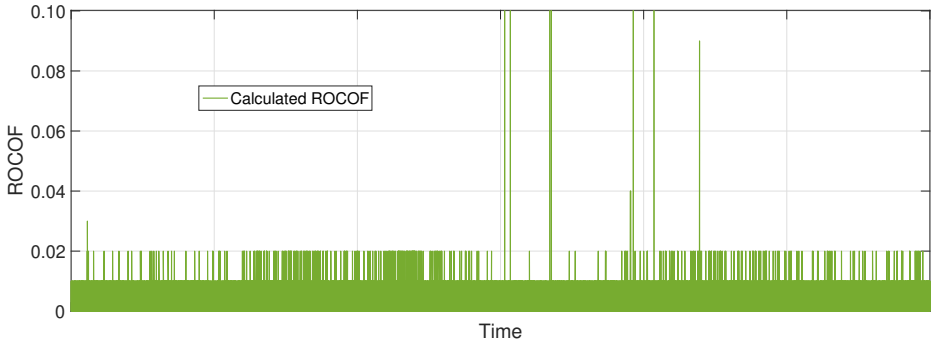


Figure B.4 Calculated ROCOF
(2017-06-20 12:28:51 to 2017-06-22 07:15:54)

```
1  void getFrequency () { // fn to calculate frequency
2    nfcmFreq = (1e6 * nfcmCount) / (2 * (nfcmEndTime - nfcmStartTime)); // calculate frequency
3    nfcmCount = 0; // reset the input signal pulse counter
4    nfcmFreq = nfcmFreq * cf * 1e3; // apply correction factor and change format (1e3)
5    // limit value of new frequency to previous value if rate of change exceeds rocofThreshold
6    // start rocof algorithm only after 4th recorded frequency after power on
7    if (rocofFlag) {
8      if (abs(freqFilter - nfcmFreq) > rocofThreshold) { // rate of change threshold
9        nfcmFreq = freqFilter; // if rocof greater than rocofThreshold set frequency
10           // to previous value
11     }
12   }
13   freqFilter = (alpha * freqFilter) + ((1 - alpha) * (nfcmFreq)); // apply filter
14   // set flag to start rocof algorithm after 4th recorded frequency after power on.
15   if (rocofCount < 4) rocofCount++;
16   if (rocofCount > 3) rocofFlag = true; // set flag to start rocof algorithm
17 }
```

Listing B.2 getFrequency()

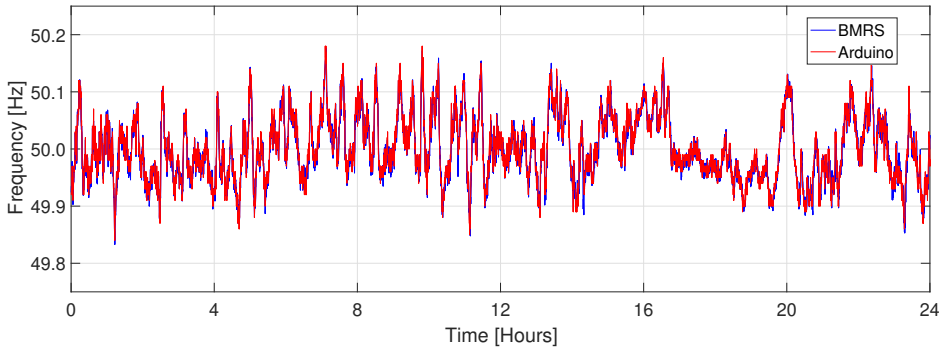


Figure B.5 Comparing frequency data
(2017-06-23 07:45:00 to 2017-06-24 07:45:00)

Appendix C

Smartphone App Development

Appendix Contents

C.1 Arduino sketch: control unit

```
1
2  /*
3  Title:        Control Unit
4  Filename:     control_unit.ino
5  Prepared by:  Sean Williams
6  Modified:     03 May 2018
7  Description:  Designed to support testing of smart phone app for thermal comfort.
8
9                Main feature include:
10               1. Support BT interface between smart phone and Arduino platform
11               2. TXRX remote temperature and humidity data via 433MHz
12               3. Interface to MATLAB/Simulink Model (USB)
13               4. Support data transfer to MVS (UART)
14               5. Ch1 provide 0-10V crtl_action cmd
15               6. Ch2 4-20mA output to LCD, brightness proportional to room_temp
16               7. LED and LCD visual feedback
17
18 32u4 > UC00A R3 > USB > PC (used for data transfer)
19 PC > 32u4 (used for *.ino loading only, not used for data transfer)
20
21 32u4      UC00A         32u4      HC5         32u4      433RXD
22 D0/RX -> TX1            D14    -> TXD         D7     -> DATA
23 D1/TX -> RX1            D15    -> RXD         +5V    -> +5V
24 +5V    -> +5V           +5V    -> +5V         GND    -> GND
25 GND    -> GND           GND    -> GND
26
27 MEGA2560 RTC            UNO       433TXD      UNO    -> DHT22
28 SDA20 -> SDA            D12    -> DATA        D4     -> 2
29 SCL21 -> SCL            +5V    -> +5V         +5V    -> 1
30 +5V    -> +5V           GND    -> GND         GND    -> 4
31 GND    -> GND
32
33 32u4 Analog 2x CH Output
34 Ch1       0-10V   ctrl_action
35 Ch2       4-20mA  LED yellow (room_temp)
36
37 32u4 Digital 8x CH Input/Output
38 Ch2       24V     LED red (smart phone: 'I'm warm')
39 Ch3       24V     LED red (smart phone: 'I'm too warm')
40 Ch4       24V     LED green (smart phone: 'I'm cold')
41 Ch5       24V     LED green (smart phone: 'I'm to cold')
42
43 Change History:
44 [05-05-2018] Attempt to use mySerial on pins D6 and D7 and connect HC05. Didnt work.
45 Confirmed with IND I/O Pin Out datasheet D6 and D7 does not support change interrupts
46 Changed D6 and D7 to D14 and D15 respectively.
47 [10-05-2018] Attempt to introduce RXD from N1 (Nano) into IND I/O.
48 Using RXDSerial (10, 11). This approach is not valid. Use instead single
49 digital pin to receive data. Nominated D7.
50 11-05-2018] Changes to data received from N2 TXD. Constraints to data before
51 output to BT, USB to UART and 32u4 LCD.
52 [10-03-2020] Changes required to support revised Smart Phone app and HIL
53 [12-03-2020] Serial.read ctrl_action 0-10V at Ch1 to EFCU-010
54 */
55
56 #include <Indio.h>
57 #include <Wire.h>
58 #include <VirtualWire.h>
59 #include <SPI.h>
60 #include <UC1701.h>
61 #include <SoftwareSerial.h>
62
63 SoftwareSerial mySerial(14, 15); //RX, TX (UC00A USB to UART converter)
64
65 const int receive_pin=7; //pin to connect 433MHz RXD data pin
66 const int redled1Pin=2;
67 const int redled2Pin=3;
68 const int grnled1Pin=4;
69 const int grnled2Pin=5;
70
71 float val = 0.0;
72 char temperatureChar[10];
73 char humidityChar[10];
74
75 struct
76 {
77   float temperature = 0.0;
78   float humidity = 0.0;
79 } data;
80
81 //typedef struct package Package;
82 //Package data;
83
84 static const byte ledPin = 13;
85
```

```
86  // The dimensions of the LCD (in pixels)...
87  //static const byte LCD_WIDTH = 128;
88  //static const byte LCD_HEIGHT = 64;
89
90  // A custom "degrees" symbol...
91  static const byte DEGREES_CHAR = 1;
92  static const byte degrees_glyph[] = {0x00, 0x07, 0x05, 0x07, 0x00};
93
94  // A custom "O" blank symbol...
95  static const byte BLANK_CHAR = 2;
96  static const byte blank_glyph[] = {0x7E, 0x42, 0x42, 0x42, 0x7E};
97
98  // A custom "O" filled symbol...
99  static const byte FILL_CHAR = 3;
100 static const byte fill_glyph[] = {0x7E, 0x7E, 0x7E, 0x7E, 0x7E};
101
102 static UC1701 lcd;
103
104 //variables
105 int i=0;
106 int BTReceived = 0;
107 int UARTReceived = 0;
108 int MATLABReceived = 0;
109 int red_state = 0;
110 int green_state = 0;
111 float MVSReceived = 0.0;
112 float anOutCh1 = 0;
113 float anOutCh2 = 0;
114 float room_temp=0;
115 float MATLABvalue=0;
116 char numStr[6];
117 String MATLABString="";
118 String inString=String();
119
120 void setup() {
121
122   lcd.begin();
123   //clears lcd screen
124   for (int y=0; y < 7; y++){
125   for (int x=0; x < 128; x++){
126     lcd.setCursor(x, y);
127     lcd.print(" ");
128     }
129   }
130
131   Serial.begin(9600);
132   //  while (!Serial) {
133   //    ; // wait for serial port to connect. Needed for native USB port only
134   //  }
135   Serial1.begin(9600);
136   mySerial.begin(9600);
137
138   // Initialise the IO and ISR
139   vw_set_rx_pin(receive_pin);
140   vw_setup(500);    // Bits per sec
141   vw_rx_start();    // Start the receiver PLL running
142
143   // Set IND.I/O CH1 and CH2 mode [10V | mA]
144   Indio.analogWriteMode(1, V10); // Set Analog-Out CH1 to 10V mode (0-10V).
145   Indio.analogWriteMode(2, mA); // Set Analog-Out CH2 to mA mode (0-20mA).
146
147   // Register the custom symbol...
148   lcd.createChar(DEGREES_CHAR, degrees_glyph);
149   lcd.createChar(BLANK_CHAR, blank_glyph);
150   lcd.createChar(FILL_CHAR, fill_glyph);
151
152   // Initial screen display
153   lcd.setCursor(0, 0);
154   lcd.print("T: ");
155   lcd.setCursor(0, 1);
156   lcd.print("H: ");
157   lcd.setCursor(0, 2);
158   lcd.print(" * Thermal Comfort *");
159   lcd.setCursor(0, 3);
160   lcd.print("CH1:    0.00V");
161   lcd.setCursor(0, 4);
162   lcd.print("CH2:    0.00mA");
163   lcd.setCursor(0, 5);
164   lcd.print("Green: ");
165   lcd.println("\002\002 ");
166   lcd.setCursor(0, 6);
167   lcd.print("Red:   ");
168   lcd.println("\002\002 ");
169
170   // Set 32u4 pinout
171   Indio.digitalMode(redled1Pin, OUTPUT);
172   Indio.digitalMode(redled2Pin, OUTPUT);
173   Indio.digitalMode(grnled1Pin, OUTPUT);
174   Indio.digitalMode(grnled2Pin, OUTPUT);
```

```
175     pinMode(ledPin, OUTPUT);
176
177     // Set Ch1 to 0V and Ch2 to 0mA
178     Indio.analogWrite(1, 0, false); //Set CH1 to 0V ("true" will write value to EEPROM ...
179         // of DAC, rstoring it after power cycle).
180     Indio.analogWrite(2, 0, false); //Set CH2 to 0mA
181
182     // Turn off all LED indicators
183     Indio.digitalWrite(redled1Pin,LOW);
184     Indio.digitalWrite(redled2Pin,LOW);
185     Indio.digitalWrite(grnled1Pin,LOW);
186     Indio.digitalWrite(grnled2Pin,LOW);
187   }
188
189   void loop() {
190
191     uint8_t buf[sizeof(data)];
192     uint8_t buflen = sizeof(data);
193     //Serial.println("149");
194     if (vw_get_message(buf, &buflen)){  // Is there a packet for us?
195       memcpy(&data,&buf,buflen);
196     //  Serial.println("153");
197     if ((data.temperature > 10.0 && data.temperature < 65) &&
198               (data.humidity > 5 && data.humidity < 60)){
199       String temperatureString = String(data.temperature,1);
200       temperatureString.toCharArray(temperatureChar,10);
201       String humidityString = String(data.humidity,1);
202       humidityString.toCharArray(humidityChar,10);
203       room_temp=data.temperature;
204       // print data to 32u4 lcd display
205       printDatalcd(temperatureString, humidityString);
206       // Write data to on board RX/TX (UC00A USB to UART Converter)
207       printDataUC00A(temperatureString, humidityString, room_temp);
208       // Write data to bluetooth (HC-05 to Android Smart Phone)
209       printDataBT(temperatureString, humidityString);
210
211       // output to Port COM (Leonardo) 23.40;B;cr/ln where B=BTReceived in {0,1,2,3,4}
212       Serial.print(data.temperature);
213       Serial.print(";");
214       Serial.print(BTReceived);
215       Serial.println(";");
216       }
217     }
218
219     // read from MATLAB ctrl_action, covert and display to LCD and set Ch1 output to
220           //same value. Data received [data_ctrl_action]=round(data_ctrl_action*100,0)
221     digitalWrite(ledPin, LOW);
222     if (Serial.available()>0){ //Serial is data from MATLAB
223       for(int i=0;i<6;++i){
224         numStr[i]=Serial.read();
225       }
226       numStr[6]='\0';
227       inString=numStr;
228       MATLABvalue=inString.toInt();
229
230       lcd.setCursor(0, 3);
231       lcd.print("CH1:    ");
232       lcd.print(MATLABvalue/100);
233       lcd.print("V      ");
234       anOutCh1=MATLABvalue/100;
235       Indio.analogWrite(1, anOutCh1, false); //Set CH1 to anOutCh1V
236           //("false" will not write value to EEPROM of DAC).
237     }
238
239     if (mySerial.available()>0){ //mySerial is data from BT
240       BTReceived = mySerial.read();
241     }
242
243     if (Serial1.available()>0){ //mySerial is data from USB-UART
244       UARTReceived = Serial1.read();
245     }
246
247     //Turn Green and Red LED OFF
248     if (BTReceived == '0'){
249       greenoff();
250       redoff();
251       Indio.digitalWrite(redled1Pin,LOW);
252       Indio.digitalWrite(redled2Pin,LOW);
253       Indio.digitalWrite(grnled1Pin,LOW);
254       Indio.digitalWrite(grnled2Pin,LOW);
255       lcd.setCursor(0, 2);
256       lcd.print("  It's warm enough  ");
257     }
258
259     //Turn on Green LED dim after pressing Cold face once
260     if (BTReceived == '1'){
261       redoff();
262       Indio.digitalWrite(redled1Pin,LOW);
```

```
263       Indio.digitalWrite(redled2Pin,LOW);
264       Indio.digitalWrite(grnled1Pin,HIGH);
265       lcd.setCursor(0, 2);
266       lcd.print("     It's cold        ");
267       lcd.setCursor(0, 5);
268       lcd.print("Green: ");
269       lcd.println("\003\002 ");
270     }
271
272     //Turn on Green LED bright after pressing Cold face twice consecutively
273     if (BTReceived == '2'){
274       Indio.digitalWrite(grnled2Pin,HIGH);
275       lcd.setCursor(0, 2);
276       lcd.print("     It's too cold    ");
277       lcd.setCursor(0, 5);
278       lcd.print("Green: ");
279       lcd.println("\003\003 ");
280     }
281
282     //Turn on Red LED dim after pressing Hot face once
283     if (BTReceived == '3'){
284       greenoff();
285       Indio.digitalWrite(grnled1Pin,LOW);
286       Indio.digitalWrite(grnled2Pin,LOW);
287       Indio.digitalWrite(redled1Pin,HIGH);
288       lcd.setCursor(0, 2);
289       lcd.print("     It's warm        ");
290       lcd.setCursor(0, 6);
291       lcd.print("Red:    ");
292       lcd.println("\003\002 ");
293     }
294
295     //Turn on Red LED dim after pressing Hot face twice consecutively
296     if (BTReceived == '4'){
297       Indio.digitalWrite(redled2Pin,HIGH);
298       lcd.setCursor(0, 2);
299       lcd.print("     It's too warm    ");
300       lcd.setCursor(0, 6);
301       lcd.print("Red:    ");
302       lcd.println("\003\003 ");
303     }
304
305   delay(1000);
306   }
307
308   void printDatalcd(String temperatureString, String humidityString){
309   // Print temperature (using the custom "degrees" symbol) and humidity data
310   // "T: 23.45'C     "
311   // "H: 22.85%      "
312     lcd.setCursor(0, 0);
313     lcd.print("T: ");
314     lcd.print(temperatureString);
315     lcd.println("\001C     ");
316     lcd.setCursor(0, 1);
317     lcd.print("H: ");
318     lcd.print(humidityString);
319     lcd.print("%     ");
320   }
321
322   void printDataUC00A(String temperatureString, String humidityString, float room_temp){
323   // Write data to on board RX/TX (UC00A USB to UART Converter)
324   // "T,23.4,H,22.8" & vbCrLf & "T,23.4,H,22.8" ...
325   // Len is 16 characters
326     anOutCh2=1+(((room_temp-15)*(18-1))/(24-15));
327     Indio.analogWrite(2, anOutCh2, false); //Set CH2 to anOuCh2mA
328             //("false" will not write value to EEPROM of DAC).
329     lcd.setCursor(0, 4);
330     lcd.print("CH2:    ");
331     lcd.print(anOutCh2);
332     lcd.print("mA    ");
333
334     Serial1.write("T,");
335     Serial1.print(temperatureString);
336     Serial1.print(",H,");
337     Serial1.println(humidityString);
338   }
339
340   void printDataBT(String temperatureString, String humidityString){
341   //Write data to bluetooth (HC-05)
342   //"23.4|22.8 & vbCrLf & 23.4|22.8" ...
343     mySerial.print(temperatureString);
344     mySerial.print("|");
345     mySerial.println(humidityString);
346   }
347
348   void greenoff(){
349     lcd.setCursor(0, 5);
350     lcd.print("Green: ");
```

```
351     lcd.println("\002\002 ");
352   }
353
354   void redoff(){
355     lcd.setCursor(0, 6);
356     lcd.print("Red:    ");
357     lcd.println("\002\002 ");
358   }
```

Listing C.1 control_unit.ino

C.2 Arduino sketch: mains frequency

```
/*
Title:          Mains Frequency
Filename:       mains_frequency.ino
Prepared by:    Sean Williams
Modified:       13 May 2018
Description     Designed to emulate Frequency Measurement tool

                Main features include:
                1. Continuous stream of loop pre-recorded mains frequency
                2. Format <yyyy-mm-ddThh:mm:ssP,ff.ff>, no carriage return

Change History
[13-May-2018] Start to add rtc functionality and array data which is an extract from
BMRS site. Nano low memoray restricing number of data items (frequency value) in array.
Uno has same capacity so stay with Nano. Invest import from txt file or introduce SD
Card data logging.
[16-05-2018] Changes output string to include flag that will be used in MVS to
indicate if test data is test harness or not.
[21-05-2018] Change output string back to previous format but make change to one element
so that can keep MVS the same except for new line that will test for this one element.
*/

#include "RTClib.h"
int secondCount;
int i = 0;
float randNumber;
//255 data entry from BMRS data, equates to 1 reading per second gives 4 minutes and 15 seconds
float freqData_1[]={
        50.01, 50.02, 50.01, 50.01, 50.01, 50.02, 50.02, 50.02, 50.00, 49.98, 49.99, 49.99,
        49.98, 50.00, 50.00, 49.99, 50.00, 50.00, 49.98, 50.00, 49.99, 49.99, 49.99, 49.98,
        49.97, 49.95, 49.96, 49.97, 49.98, 49.96, 49.96, 49.97, 49.96, 49.97, 49.98, 49.97,
        49.96, 49.97, 49.96, 49.98, 49.99, 49.97, 49.97, 49.99, 49.98, 49.97, 49.97, 49.97,
        49.97, 49.98, 49.99, 49.99, 50.00, 49.97, 49.98, 49.98, 49.98, 49.98, 49.98, 49.97,
        49.98, 49.96, 49.98, 49.97, 49.98, 49.97, 49.96, 49.95, 49.95, 49.95, 49.96, 49.94,
        49.94, 49.96, 49.95, 49.95, 49.94, 49.95, 49.96, 49.96, 49.96, 49.98, 49.97, 49.96,
        49.95, 49.94, 49.94, 49.95, 49.95, 49.95, 49.92, 49.93, 49.94, 49.95, 49.95, 49.93,
        49.93, 49.95, 49.96, 49.95, 49.96, 49.97, 49.97, 49.97, 49.96, 49.95, 49.95, 49.97,
        49.96, 49.96, 49.97, 49.96, 49.97, 49.94, 49.94, 49.96, 49.97, 49.98, 49.96, 49.98,
        49.97, 49.97, 49.97, 49.96, 49.98, 49.99, 49.98, 49.97, 49.99, 49.99, 49.99, 50.01,
        49.98, 49.98, 49.99, 49.99, 49.99, 49.99, 50.00, 50.02, 50.02, 50.03, 50.03, 50.00,
        50.00, 49.99, 50.02, 50.02, 50.01, 50.01, 50.01, 50.03, 50.04, 50.06, 50.06, 50.08,
        50.10, 50.09, 50.09, 50.10, 50.12, 50.11, 50.09, 50.08, 50.08, 50.06, 50.05, 50.03,
        50.04, 50.05, 50.04, 50.02, 50.00, 49.99, 50.00, 49.99, 49.97, 49.95, 49.93, 49.94,
        49.93, 49.97, 49.94, 49.92, 49.93, 49.93, 49.92, 49.90, 49.91, 49.92, 49.93, 49.93,
        49.94, 49.94, 49.94, 49.93, 49.93, 49.91, 49.93, 49.93, 49.95, 49.97, 49.95, 49.96,
        49.96, 49.98, 49.96, 49.97, 49.98, 49.99, 49.98, 49.98, 49.99, 49.99, 50.02, 50.00,
        49.99, 49.98, 49.99, 49.98, 50.01, 49.99, 49.98, 50.03, 50.02, 50.03, 50.02, 50.02,
        50.01, 50.00, 50.02, 50.02, 50.02, 50.03, 50.04, 50.01, 50.02, 50.01, 50.01, 50.00,
        50.03, 50.02, 50.01, 50.02, 50.02, 50.03, 50.02, 50.03, 50.03, 50.04, 50.02, 50.02,
        50.05, 50.04, 50.03, 50.01, 50.01, 50.01, 50.03, 50.02, 50.03, 50.05, 50.05, 50.05,
        50.06, 50.05, 50.03, 50.05, 50.04, 50.05, 50.03, 50.04, 50.06, 50.06, 50.05, 50.04,
        50.03, 50.02, 50.04, 50.06, 50.06, 50.06, 50.06, 50.07, 50.06, 50.07, 50.05, 50.05,
        50.07, 50.09, 50.09, 50.10, 50.10, 50.10, 50.09, 50.08, 50.08, 50.07, 50.07, 50.09,
        50.09, 50.08, 50.08, 50.07, 50.07, 50.05, 50.02, 50.02, 50.03, 50.05, 50.06, 50.06,
        50.06, 50.08, 50.08, 50.08, 50.10, 50.08, 50.06, 50.10, 50.10, 50.08, 50.10, 50.10,
        50.11, 50.11, 50.11, 50.09, 50.06, 50.07, 50.07, 50.09, 50.10, 50.10, 50.10, 50.09,
        50.13, 50.11, 50.10, 50.09, 50.08, 50.09, 50.07, 50.10, 50.11, 50.11, 50.10, 50.10,
        50.07, 50.06, 50.08, 50.04, 50.05, 50.04, 50.03, 50.03, 50.05, 50.04, 50.04, 50.02,
        50.01, 50.03, 50.01, 49.99, 50.01, 50.01, 49.97, 49.99, 49.98, 49.97, 49.98, 49.97,
        49.96, 49.94, 49.94, 49.96, 49.95, 49.95, 49.97, 49.95, 49.93, 49.95, 49.94, 49.93,
        49.94, 49.94, 49.95, 49.95, 49.93, 49.92, 49.93, 49.92, 49.91, 49.92, 49.92, 49.92,
        49.92, 49.94, 49.92, 49.92, 49.92, 49.90, 49.91, 49.92, 49.93, 49.91, 49.92, 49.92,
        49.90, 49.90, 49.90, 49.88, 49.88, 49.86, 49.85, 49.84, 49.86, 49.84, 49.86, 49.91,
        49.95, 49.95, 49.96, 49.96, 49.95, 49.96, 49.96, 49.96, 49.98, 49.98, 49.99, 50.00,
        50.02, 50.02, 50.03, 50.04, 50.05, 50.05, 50.04, 50.05, 50.05, 50.07, 50.06, 50.07,
        50.08, 50.05, 50.05, 50.07, 50.08, 50.08, 50.07, 50.08, 50.10, 50.07, 50.06, 50.04,
        50.06, 50.07, 50.07, 50.06, 50.06, 50.07, 50.05, 50.06, 50.06, 50.06, 50.06, 50.04,
        50.04, 50.05, 50.06, 50.10, 50.07, 50.07, 50.05, 50.03, 50.04, 50.03, 50.02, 50.02,
        50.02, 50.01, 50.02, 50.02, 50.03, 50.01, 50.00, 49.99, 49.99, 49.98, 49.99, 49.99,
        50.01, 50.03, 50.01, 50.01, 50.00, 49.99, 49.99, 49.99, 50.02, 50.00, 50.02, 50.02,
        50.05, 50.03, 50.03, 50.01, 50.02, 50.02, 50.00, 50.02, 50.01, 50.02, 50.01, 50.02,
        50.03, 50.01, 50.02, 50.01, 50.02, 50.01, 50.00, 50.01, 50.03, 50.03, 50.02, 50.00,
        49.99, 49.99, 50.00, 50.03, 50.01, 50.03, 50.01, 50.03, 50.03, 50.02, 49.99, 50.00,
        49.99, 50.01, 50.02, 50.04, 50.02, 50.04, 50.04, 50.04, 50.05, 50.06, 50.03, 50.06,
        50.06, 50.07, 50.08, 50.09, 50.08, 50.06, 50.09, 50.09, 50.07, 50.04, 50.04, 50.04,
        50.06, 50.05, 50.06, 50.05, 50.07, 50.05, 50.01, 50.01, 49.99, 50.00, 50.02, 50.01,
        50.02, 50.02, 50.02, 50.04, 50.01, 50.01, 50.00, 50.01, 50.02, 50.02, 49.98, 49.99,
        50.00, 49.99, 49.97, 49.97, 49.97, 49.97, 49.96, 49.98, 49.96, 49.95, 49.97, 49.98,
        49.97, 49.97, 49.94, 49.95, 49.95, 49.94, 49.95, 49.93, 49.94, 49.94, 49.93, 49.93,
        49.90, 49.92, 49.92, 49.93, 49.94, 49.94, 49.95, 49.96, 49.96, 49.96, 49.93, 49.91,
        49.92, 49.93, 49.92, 49.90, 49.89, 49.88, 49.88, 49.89, 49.89, 49.93, 49.96, 49.95,
        49.96, 49.96, 49.95, 49.96, 49.96, 49.98, 49.96, 49.96, 49.98, 49.97, 49.98, 49.97,
        49.97, 49.97, 49.96, 49.96, 49.96, 49.95, 49.94, 49.93, 49.93, 49.92, 49.93, 49.93,
        49.94, 49.92, 49.91, 49.92, 49.92, 49.89, 49.90, 49.92, 49.91, 49.93, 49.94, 49.95,
```

```
85          49.94, 49.95, 49.93, 49.94, 49.95, 49.96, 49.96, 49.97, 49.97, 49.95, 49.95, 49.98,
86          49.97, 49.98, 49.99, 49.99, 49.96, 50.00, 50.03, 50.05, 50.06, 50.07, 50.06, 50.07,
87          50.06, 50.06, 50.07, 50.06, 50.08, 50.07, 50.07, 50.07, 50.05, 50.06, 50.09, 50.07,
88          50.08, 50.10, 50.08, 50.06, 50.06, 50.05, 50.04, 50.02, 50.00, 49.99, 50.01, 50.02};
89
90  int dataCount = sizeof freqData_1/sizeof freqData_1[0]; // get size of test array freqData_1
91
92  String dataStringA, dataStringB, dataStringC; // place holders for data string
93                                                // dataStringA = <yyyy-mm-ddT
94                                                // dataStringB = hh:mm:ssP,
95                                                // dataStringC = ff.ff>
96  String stringYear, stringMonth, stringDay; //dataStringA
97  String stringHour, stringMinute, stringSeconds, stringMilliSeconds; //dataStringB
98  String stringFreq; //dataStringC
99
100 RTC_DS1307 rtc;
101
102 char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
103         "Thursday", "Friday", "Saturday"};
104
105 void setup() {
106
107   Serial.begin(9600);
108     if (! rtc.begin()) {
109     Serial.println("Couldn't find RTC");
110     while (1);
111   }
112
113   if (! rtc.isrunning()) {
114     //Serial.println("RTC is NOT running!");
115     // following line sets the RTC to the date & time this sketch was compiled
116     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
117     // This line sets the RTC with an explicit date & time, for example to set
118     // January 21, 2014 at 3am you would call:
119     // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
120   }
121
122 }
123
124 void loop() {
125
126   randNumber = (float (rand())/float((RAND_MAX))*2)+49;
127   getDataString(); // fn to format date time group and frequency
128
129   Serial.print(dataStringA);
130   Serial.print(dataStringB);
131   Serial.print(dataStringC);
132   delay(1000);
133 }
134
135 void getDataString() { // fn to set data into correct format
136     DateTime now = rtc.now();
137   // dataString defined by dataStringA, dataStringB and dataStringC
138
139   // dataStringA: <yyyy-mm-ddT
140   stringYear = String(now.year());
141   stringMonth = now.month();
142   if (stringMonth.length() == 1) stringMonth = "0" +stringMonth;
143   stringDay = now.day();
144   if (stringDay.length() == 1) stringDay = "0" +stringDay;
145   dataStringA = "";
146   dataStringA = "<";
147   dataStringA += stringYear;
148   dataStringA += "-";
149   dataStringA += stringMonth;
150   dataStringA += "-";
151   dataStringA += stringDay;
152   dataStringA += "T";
153
154   // dataStringB: hh:mm:ssP,     'P' replaces 'Z' in this version
155   // ss is set in pps and timer1
156   stringHour = now.hour();
157   if(stringHour.length() == 1) stringHour = "0" + stringHour;
158   stringMinute = now.minute();
159   if(stringMinute.length() == 1) stringMinute = "0" + stringMinute;
160   stringSeconds = now.second();
161   if(stringSeconds.length() == 1) stringSeconds = "0" + stringSeconds;
162   dataStringB = "";
163   dataStringB += stringHour; dataStringB += ":";
164   dataStringB += stringMinute; dataStringB += ":";
165   dataStringB += stringSeconds; dataStringB += "P,";
166   dataStringC = "";
167
168   dataStringC = String(freqData_1[i]) + dataStringC + String(">");
169   i++;
170   if (i > dataCount-1) i=0;
171 }
```

Listing C.2 mains_frequency.ino

C.3 Arduino sketch: transmitter

```
/*
Title:          Transmitter
Filename:       transmitter.ino
Prepared by:    Sean Williams
Modified:       16 May 2018
Description     Designed to TX data from remote node to master.

                Main features include:
                1. Continuous stream of temperature and humidity data
                2. Sample rate 2sec
                3. Compatable for Nano or Uno

Change History

*/

#include <DHT.h>
#include <VirtualWire.h>

#define DHTPIN 4
#define DHTTYPE DHT22

const int led_pin = 13;
const int transmit_pin = 12;

struct package
{
  float temperature ;
  float humidity ;
};

typedef struct package Package;
Package data;

DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  // Initialise the IO and ISR
  vw_set_tx_pin(transmit_pin);
  vw_set_ptt_inverted(true); // Required for DR3100
  vw_setup(500);        // Bits per sec
  pinMode(led_pin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Transmitter");
}

void loop()
{
  readSensor();
  if ((data.temperature > 10) && (data.humidity >10)){
  digitalWrite(led_pin, HIGH); // Flash a light to show transmitting
  vw_send((uint8_t *)&data, sizeof(data));
  vw_wait_tx(); // Wait until the whole message is gone
  digitalWrite(led_pin, LOW);
  delay(2000);
  }
}

void readSensor()
{
  dht.begin();
  delay(1000);
  data.humidity = dht.readHumidity();
  data.temperature = dht.readTemperature();
  Serial.print(data.temperature);
  Serial.print(" ");
  Serial.println(data.humidity);
}
```

Listing C.3 transmitter.ino

C.4 MIT App Inventor 2 Block Code

Figure C.1 App block code: initialisation

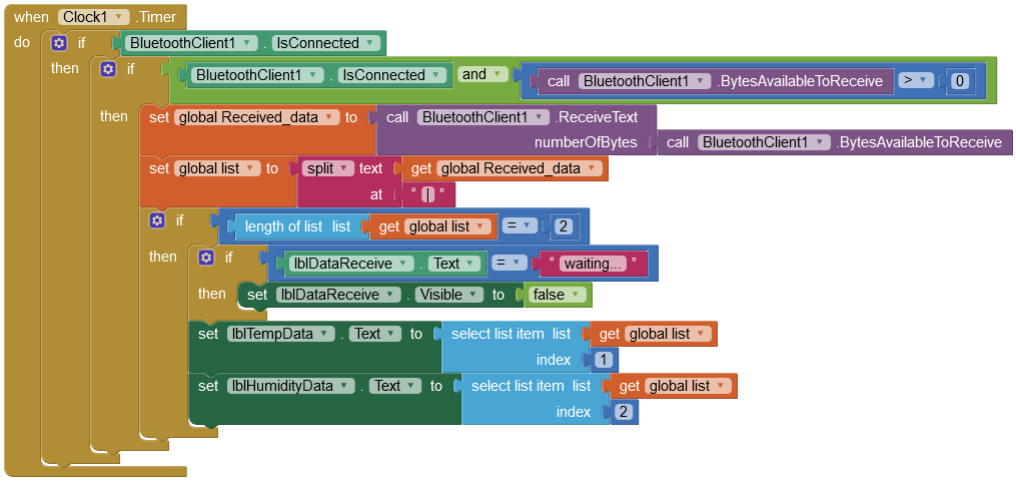


Figure C.2 App block code: interaction

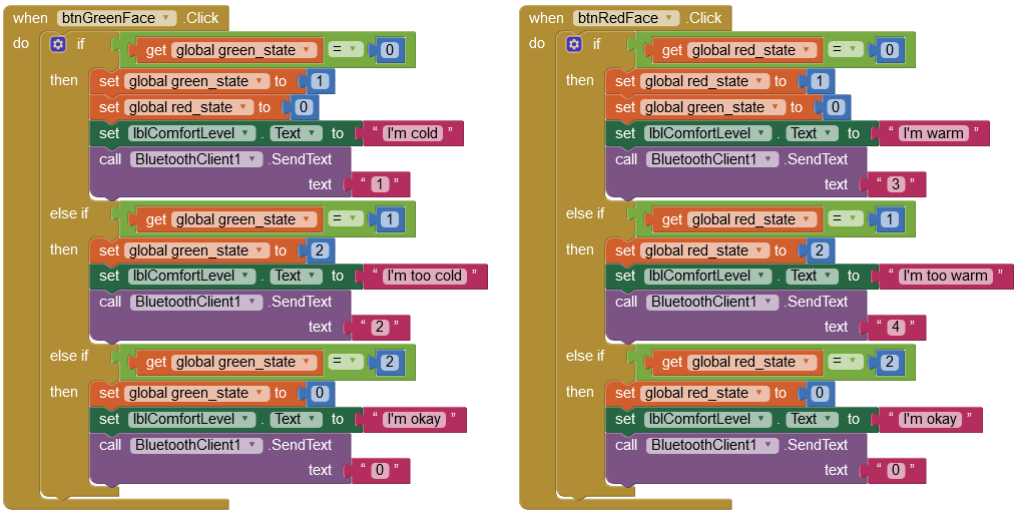


Figure C.3 App block code: data

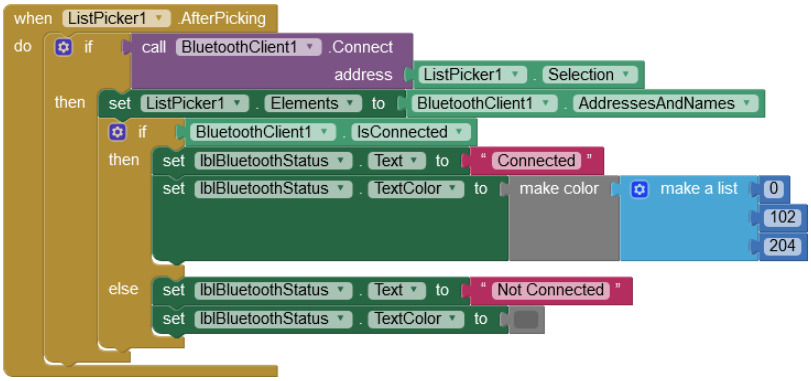


Figure C.4 App block code: communication

C.5 Hardware-in-the-loop test wiring diagram

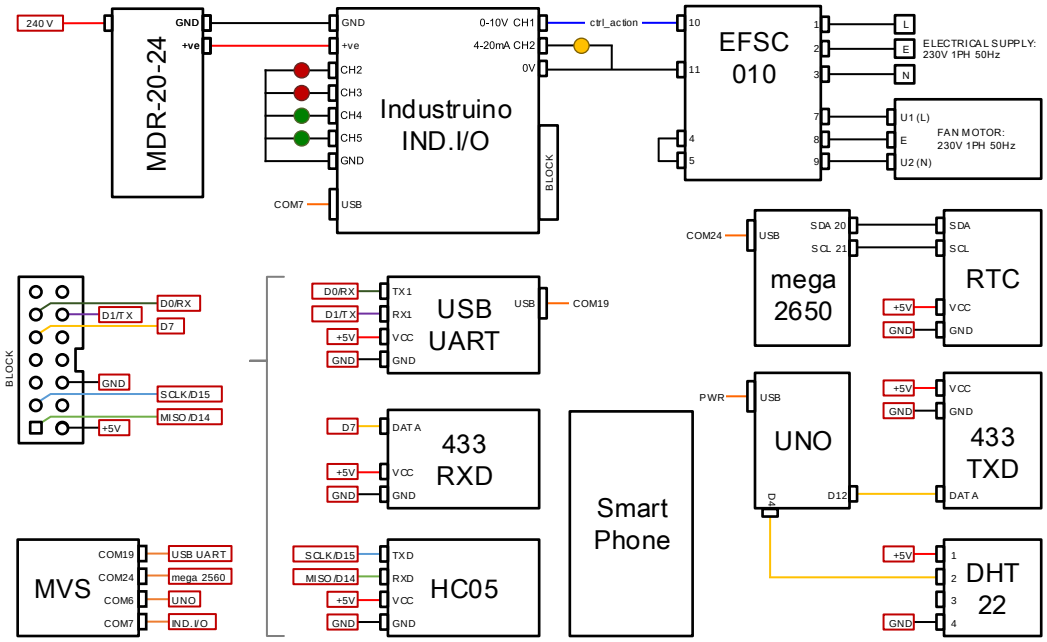


Figure C.5 Hardware-in-the-loop test wiring diagram

C.6 Simulation model code updates

In addition to the changes made to the Simulink® model block layout, several software code updates and new functions are required. The table below summarises the relevant code changes, indicating if the code has been updated (U) or if it is a new (N) addition to the catalogue of existing code listings.

Table C.1
HIL software code

Code Name	Status	Description
optim_ctrl.m	U	New input port `InputPort(2).Data` for `S1tcv` `S1tcv` variable is passed to function `prepare_tc_gridmap.m`
optim_ctrl_model_data.m	U	The following sections have been deleted: • Set Outdoor Temperature Variation • Set Building Parameters • Set Power Systems Parameters Modified Set Date Time: • Delete references to `daily_temp` • Include `ifelse` statement at end of section Case 1-6 `date_time` remains unchanged
prepare_tc_gridmap.m	U	The following code changes have been documented: • Code change enables signal from single smartphone to interact with Simulink model • `tc` at S1 is set to feedback from smartphone irrespective of planned occupancy. Code that sets response for S2 to S24 remains unchanged • Include S1tcv as input parameter • `tcv(3)` (calc_mode) at S1 set to `S1tcv` (user themal comfort feedback)
read_serialdata.m	N	Function reads room temperature and thermal comfort from serial port
readdata.m	N	Room temperature and thermal comfort routed from Industruino using USB connected to serial port with room temperature from remote Arduino Uno and DHT22 sensor using wireless connection and thermal comfort from Android smartphone using Bluetooth connection Sampling time is set to 5 min (300 sec)

continued . . .

. . . continued

Code Name	Status	Description
soc.m	U	The following code changes have been documented: • Delete all references to `path_2` • Building subsystem removed from Simulink® model
te2u.m	N	Function sets temperature setpoint (control action) depending on measured temperature. System limited to operate in temperature range 15.5 °C (minimum) to 20.5 °C (maximum)
write_serialdata.m	N	Enable data transfer `data_ctrl_action` parameter is multiplied by 100 and rounded before TX Industruino RX divide by 100 to restore value
writedata.m	N	Write control action value to serial port. Sample rate is 5 min (300 sec)

Appendix D

Energy Management Technical Development

Appendix Contents

D.1 Simulink model: energy subsystem

Decentralised DR frequency regulation, when used in building stock, can regulate short-term frequency excursions in demanded electrical energy [5]. The contribution of a decentralised frequency regulator has been analysed [5]. Results presented suggest that small excursions in measured temperature from a TCL setpoint value will not compromise indoor comfort temperatures but can contribute to the restoration of frequency equilibrium during network stress events. In this chapter, we integrate the implied linear power system and frequency regulator as part of the optimise and control framework. The model (energy_subsystem) shown in Figure D.1 replicates a power system rating of 300 MVA. Initial conditions assume the balance in supply and demand is at equilibrium, measured frequency is 50 Hz and the steady-state frequency error is zero. The energy subsystem model parameters are reported in Table D.1.

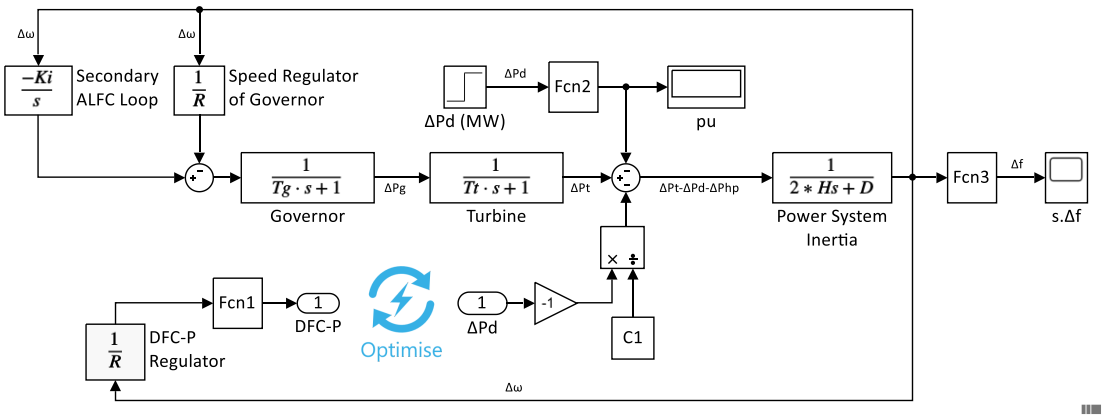


Figure D.1 Simulink® model of energy subsystem

Table D.1
Energy model parameters

Parameter	Description	Value
Ki	Secondary ALFC integral gain	1.667e-3
R	Governor speed regulator	0.05 Hz/pu MW
Tg	Governor time constant	0.25 sec
Tt	Turbine time constant	0.60 sec
H	Inertia time constant	5 sec
$D1$	Load damping constant	0.8 sec
C1	Constant	10e6
ΔPd	Contingency load	75 MW

D.2 Simulink model: building system

The building subsystem model (building_subsystem) shown in Figure D.2, is a simplified thermostatically controlled (on/off) heating system with feedback loops which typically maintains the air temperature at a set level. The model emulates building thermodynamics (building), calculating variations in temperature based on heat flow, $H(t)$, and heat losses, $H_{loss}(t)$.

$$H_{loss}(t) = \frac{T_{room} - T_{out}}{R_{th}} \tag{D.1}$$

$$\frac{\Delta T_{heater}}{\Delta t} = \frac{1}{\dot{M}c}\left(H(t) - H_{loss}(t)\right) \tag{D.2}$$

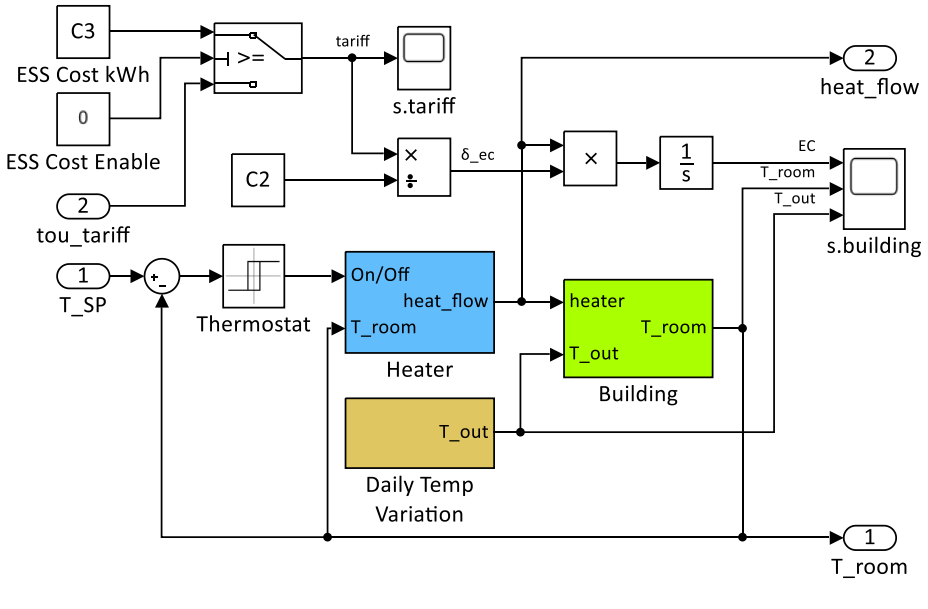
A series of embedded lookup tables representative of seasonal variation are used to model outside air temperature over a 24 hr period at a sample rate of 30 min [242]. In practice, the local outdoor temperature is measured using sensors and input into the system. Energy cost (EC [p/kWh]) is calculated as a function of time and heat flow and is expressed in the following equation:

$$EC = \int_{t_0}^{t_n} \left((T_{heater} - T_{room})\dot{M}c\right) \delta_{\mathsf{ec}}(x_{t(n)}) \tag{D.3}$$

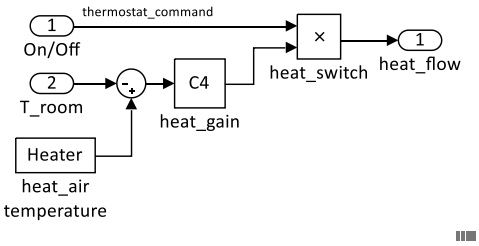
Where $\dot{M}$ [kg/hr] denotes mass air flow rate through the heater; $c = 1005.4$ is the specific heat at constant air pressure and, $\delta_{\mathsf{ec}}(x_{t(n)})$ [p/kWh] is the energy price at time $t(n)$. The building subsystem model parameters are reported in Table D.2.

Table D.2
Building model parameters

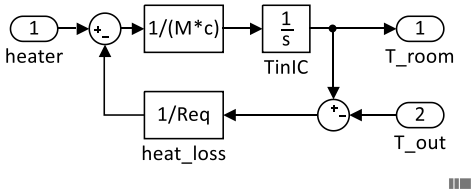
Parameter	Value
C2	3.6e3
C3	0.0199
C7	15
C8	1800



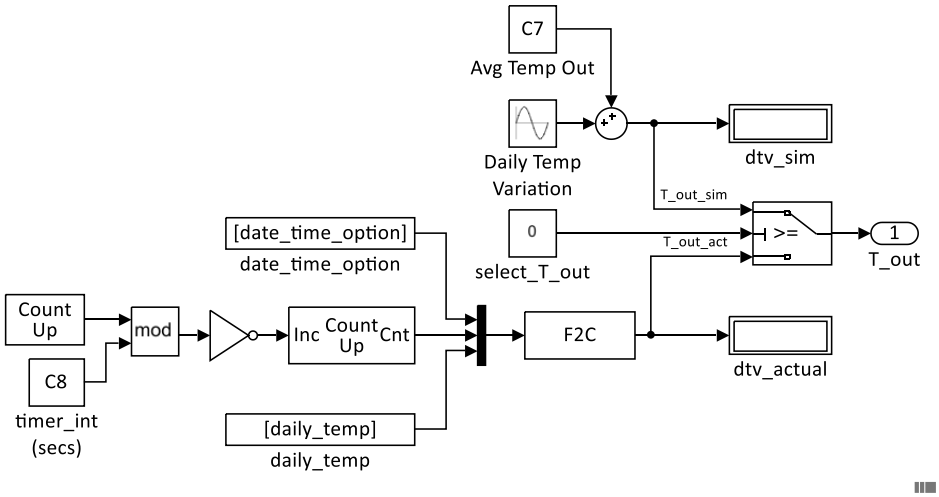
(a) Building subsystem



(b) Heater



(c) Building



(d) Daily temperature variation

Figure D.2 Simulink® model of building subsystem

D.3 Piecewise function worked example

A piecewise function has been developed to calculate a demand value at any time over a 24-h period (weekday or weekend day). The function `demand.m` has been configured to calculate demand values over a 24-h period by repeatedly using the piecewise function $S(x)$. The following table details parameters for weekday cubic spline interpolation only.

Table D.3
Piecewise function weekday data

Start	Stop	Si	lo	hi	ai	bi	ci	di	X	Y
00:00	01:00	0	0	2	15.344	0.000	0.241	-0.120	2	15.34
01:00	03:00	1	2	6	15.344	-0.482	-0.482	0.074	6	10.47
03:00	05:00	2	6	10	10.470	-0.763	0.412	0.156	10	24.00
05:00	07:00	3	10	14	24.002	10.028	2.286	-0.368	14	77.12
07:00	09:00	4	14	18	77.116	10.634	-2.135	0.163	18	95.94
09:00	11:00	5	18	22	95.942	1.391	-0.176	-0.010	22	98.02
11:00	13:00	6	22	26	98.022	-0.517	-0.301	0.044	26	93.99
13:00	15:00	7	26	30	93.986	-0.790	0.233	0.002	30	94.65
15:00	17:00	8	30	34	94.648	1.145	0.251	-0.101	34	96.80
17:00	19:00	9	34	38	96.800	-1.682	-0.958	0.152	38	84.47
19:00	21:00	10	38	42	84.466	-2.056	0.864	-0.262	41	79.01
21:00	23:00	11	42	46	73.323	-7.703	-2.276	0.470	42	73.32
23:00	00:00	12	46	48	36.162	-3.361	3.361	-0.840	48	36.16

The centre point of each PAA segment defines a set of evenly spaced nodes. The piecewise function $S(x)$ interpolates all local data points and hence confines the ill-effects of any erroneous data points, Equation (D.4).

$$S_i(x) = a_i + b_i(x - i_{lo}) + c_i(x - i_{lo})^2 + d_i(x - i_{lo})^3 \tag{D.4}$$

Where $i \in [0, 1, \ldots, n]; x \in [lo, hi]$; where lo and hi define the start and end data points of each PAA segment, respectively. The cubic polynomial coefficients are represented by the parameters a_i, b_i, c_i and d_i (Table D.3).

To calculate demand value at 20:30h for weekday,

$$S_{10}(41) = 84.466 - 2.056(41 - 38) + 0.864(41 - 38)^2 - 0.262(41 - 38)^3 = 79.01 \tag{D.5}$$

where $x = 41$ is equivalent to 20:30h.

D.4 A note about MATLAB® and Simulink®

The MATLAB® and Simulink® platform [185] is optimised for solving complex engineering problems. It has been the tool of choice throughout this research study. The vast library of pre-built toolsets has enabled efforts to focus on broader issues relating to the chosen subject. Where pre-built toolsets are not available, then custom blocks with multiple input ports and output ports capable of handling any signal produced by a Simulink® model have been created. Level-2 MATLAB S-functions with callback methods defines the properties and behaviour of custom blocks. The MATLAB® and Simulink® release used during this research is R2018b (9.5.0.944444).

D.5 Computer simulation model: function description

Table D.4
Computer simulation model: function description

Function	Description
comfort2.m	Computes thermal comfort value to minimise the objective function. **Construction:** `[O,R,M]=COMFORT(DATE,INFO[N])` for a given time `O` is the number of occupants, `R` is the number of responses and `M` is the mode comfort values that is dependent on number of occupants. The number of occupants is between a minimum and maximum threshold depending on the time of day. Mode is calculated only if number of occupants exceeds minimum response threshold. Occupant response is biased depending on representative change in outdoor temperature (summer profile). Option to display comfort information is set by logic operator `INFO [TRUE]/False`.
date2sec.m	Rounds date and time to nearest second. **Construction:** `[YVEC]=DATE2SEC(Y)` rounds date and time to nearest second. The output `Y` is serial data number.
date2vec.m	Converts date and time to vector components. **Construction:** `[YVEC]=DATE2VEC(Y)` converts data and time to vector components. The output represents the date and time components of hours, minutes, seconds, day, month, and year. The output `Y` is serial data number.

continued ...

. . . continued

Function	Description
demand.m	Computes demand value to minimise the objective function. **Construction:** `DEMAND(DATE,INFO[N]` uses cubic spline coefficients to compute demand value based on analysis of chronological sequence of discrete observations. Polynomial coefficient structure in computed directly from observations after piecewise aggregated approximation has been applied. Separate structures for weekday and weekend day are prepared. Monthly variations (seasonal adjustments) are applied using PAA Lookup Table (LUT). The demand value returned has been optimised (rescaled) for use with an Energy Optimisation System (EOS). The option to display demand information (including plot) is set by logic operator `INFO [TRUE]/False`.
demo_tdv.m	Provides option to use outdoor temperature variation. **Construction:** `[MEASURED_TEMP]=DEMO_DTV(DATA)` returns measured temperature variation in degC for selected data. Where: `DATA(1,1)`=date_time_option indicator. `DATA(2,1)`=count increments every Timer_inc (sec) [1800]. `DATA(2:49,1)`=measured outdoor temperature variation for selected date_time.
dijkstra.m	Computes a gridmap cost and shortest path. **Construction:** `[COST,PATH]=DIJKSTRA(G,S,T)` computes the cost and shortest path starting at node `S` and ending at node `T`. Where `G` is a weighted graph (digraph) that has been converted initially into a sparse adjacency matrix then into a full matrix. The PATH contains all the nodes on the shortest path. The weights between each node on the shortest path are returned as a single value representing the total cost from node `S` to node `T`.

continued . . .

. . . continued

Function	Description
initialise.m	Initialise parameters that set visual mode, horizon window and demand event duration. Function also loads gridmap template from compressed MAT-file, which contains both 1-D and multidimensional array. Default settings: Horizon window = 4 (4 hr). Visual mode = 1 (none). Demand event duration = 2 (40 min). Temperature path = 3 (Troom plus 2 °C).
optim_ctrl.m	Code tagged to Simulink model block optimise_subsystem. On receipt of date/time (sample rate: 10 min) code computes new temperature setpoint `ctrl_action` using Dijsktra's algorithm which is a function of occupant thermal comfort, electricity demand and cost (tariff). Code reacts on receipt of demand event signal.
optim_ctrl_model_data.m	MATLAB function includes option to select simulated daily temperature variation or measured daily temperature. In addition, parameters for power system (energy_subsystem) and building (building_subsystem) are defined.
prepare_comfort_values.m	MATLAB function sets temperature setpoint (control action) depending on measured temperature. System limited to operate in temperature range 15.5 °C (minimum) to 20.5 °C (maximum).
prepare_digraph.m	MATLAB function prepares digraph, transposing gridmap (31×72) to edgelist (733×3) before creating sparse adjacency matrix and finally full adjacency matrix. The start and end nodes and their respective edge weights format is prepared for fcn dijkstra.
prepare_dv_values.m	MATLAB function computes demand values for duration of horizon window at sample rate of 10 min; one per stage. Formating for scrolling figure including rescaled, gridmap and nodepath included.
prepare_edgepath.m	MATLAB function returns list of numbers that describes edgepath between start and end nodes of each stage. For each start and end node pair code searches for index from nodemap (11×25) at each stage. The intercept is the edgepath; starting from `S1` to `S24`.

continued . . .

... continued

Function	Description
prepare_gridmap.m	MATLAB function starts with gridmap $(31 \times 72 \times 4)$ template. Maps nodepath (11×25) onto gridmap (31×72) for each objective function (page): comfort, demand and tou (tariff). At each stage, the min value is defined as the stage centroid, all remaining values are populated, increasing/decreasing in value moving up/down in the same col (stage). The index where the min value at `t0` and `t240` is found are stored in `t0minidx` and `t240minidx` respectively. Exception handling at boundary upper and lower is included.
prepare_tc_gridmap.m	MATLAB function starts with gridmap $(31 \times 72 \times 4)$ template. Maps nodepath (11×25) onto gridmap (31×72) for each objective function (page): comfort, demand and tou (tariff). At each stage, the min value is defined as the stage centroid, all remaining values are populated, increasing/decreasing in value moving up/down in the same col (stage). The index where the min value at `t0` and `t240` is found are stored in `t0minidx` and `t240minidx` respectively. Exception handling at boundary upper and lower is included. Similar to `prepare_gridmap_.m` but specific to thermal comfort.
prepare_tou_values.m	MATLAB function computes tou value set for `S1` to `S24` at every 10 min interval specific to tou tariff and time of day.

continued ...

. . . continued

Function	Description
soc.m	Controls switching of energy storage asset and grid power mode of operation. Operates in 2 modes: [1] Normal operations. [2] Demand event. Assume normal operations (MODE=0). Initially SOC assumed 0 and will start to charge. At high threshold ESS declared available for use (FIT=1). FIT status revert to 0 when low threshold reached (on discharge). If SOC available and tariff HIGH (level 3), power switch to ESS (PWR=1). When tariff LOW (level 1 or 2) power switch to GRID (PWR=0). On receipt of demand event signal, MODE=1. Priority sets ESS to charge during 4 hr ramp time before demand event starts and power switch to GRID (PWR=0). At demand event start power switch to ESS (PWR=1), ESS begins to discharge. Maintain ESS power for duration of demand event. At end of demand event revert to normal operations (MODE=0). Self-Discharge Rate (SDR) applies on discharge.
tariff_mode.m	Computes tariff mode value. **Construction:** `[T_MODE]=TARIFF_MODE(TARIFF)` compares tariff at given time of day against set criteria. Model set criteria includes two levels: Level 1: 4.99 Level 2: 11.99 If tariff is greater than zero and less than or equal to Level 1 then set the tariff mode to 1 (t_mode=1). If tariff is greater than Level 1 and less than or equal to Level 2 then set the tariff mode to 2 (t_mode=2). If tariff is greater than Level 2 (assumed to be highest tariff band) then set the tariff mode to 3 (t_mode=3).

D.6 comfort_2.m

```
function [cv] = comfort_2(cn_date,info)
%%COMFORT Computes thermal comfort value to minimise the objective function.
%
%  Construction:
%   [O, R, M] = COMFORT(DATE,INFO[N]) for given time O is the number of
%   occupants, R is the number of responses and M is the mode comfort value
%   that is dependant on number of occupants.
%
%   The total number of occupants is between a minimum and maximum
%   threshold depending on time of day. Mode is calculated only if number
%   of response from total number of occupants exceeds minimum response
%   threshold. Occupant response is biased depending on representative
%   change in outdoor temperature (summer profile).
%
%   Option to display comfort information is set by logic operator INFO
%   [TRUE]/False,
%
%   Example (Publish):
%
%   >> comfort=comfort_2('12-Nov-2019 13:30:00,1)
%
%           maxoccupants: 70
%              occupants: 15
%               response: 12
%       response_percent: 80
%     response_threshold: 40
%     threshold_exceeded: 'Yes'
%    outdoor_temperature: 4
%                     cn: 13:30:00
%                    HN: 4
%                     N: 3
%                     Z: 0
%                     P: 5
%                    HP: 0
%              calc_mean: -0.5000
%            calc_median: -1
%              calc_mode: 1
%
%   comfort = 70 12 -1
%
%   where    15 = number of occupants
%            12 = number of responses
%            -1 = thermal comfort (mode)

%% Additional Information
%
%  Weekday Profile
%  Time      Period  u1   u2   u4          v
%  00h-09h      1     0    0    1          0 1 2
%  09h-11h      2    10   40    2       -1 0 1 2
%  11h-13h      3     5   20    3       -1 0 1
%  13h=15h      4    15   70    4    -2 -1 0 1 2
%  15h-17h      5     3   12    5    -2 -1 0
%  17h-19h      6     7   30    3       -1 0 1
%  19h-24h      7     0    0    2       -1 0 1 2
%
%  Weekend Profile
%  Time      Period  u1   u2   u4          v
%  00h-24h      1     0    0    1          0 1 2
%
%  u1=minimum occupancy
%  u2=maximum occupancy
%  u4=relative outdoor temperature
%  v=bias range
%

% Check number of inputs.
if nargin > 2
    error('myfuns:nd:TooManyInputs', ...
        'requires at most 2 inputs');
end

% Fill in unset optional values.
switch nargin
    case 1
        info = 0;
```

```
77  end
78
79  %% MATLAB Function Description
80  %
81  %  Title: Thermal Comfort
82  %  Filename: comfort_2.m
83  %  Prepared by: Sean Williams
84  %  Date: 24-10-2019
85  %
86  %  MATLAB function computes thermal comfort value for Energy Control
87  %  and Optimisation Framework (ECOF).
88  %
89
90  %% Change History
91  %
92  % # [06-08-2019] Initial.
93  % # [06-09-2019] Introduce option to determine if zero occupancy at time
94  %   now plus cn_date minutes.
95  % # [24-10-2019] Replace stime cariable to set datetime, instead use
96  %   S0_date and Sn_date combination inline with demand and tou functions.
97  %   Additional change implemented that checks if date time is weekend or
98  %   weekday. Refer to Addition Information for weekend and weekday
99  %   occupancy profile. Assumes no planned occupancy at weekends.
100 %
101
102 %% Initialise Variables
103
104 % format function input date time
105 ccn_date=datetime(cn_date,'ConvertFrom','datenum');
106
107 % extract time of day from cnn_date
108 cn=timeofday(ccn_date);
109
110 % minimum number of occupants for each period (1-7)
111 u1=[0 10 5 15 3 7 0];
112
113 % maximum number of occupants for each period (1-7)
114 u2=[0 40 20 70 12 30 0];
115
116 % minimum number of forced responses for each period (1-7)
117 u3=ceil(u1.*0.5);
118
119 % relative outdoor temperature (summer) range
120 u4=[1 2 3 4 5];
121
122 % set minimum number of responses required before responses are effective
123 responsethreshold=40;
124
125 % set relative outdoor temperature bias
126 v1=0:2;
127 v2=-1:2;
128 v3=-2:1;
129
130 % check if weekend/week day (refer to Additional Information for occupancy
131 % profile
132 if (isweekend(ccn_date))
133     minoccupant=u1(1);
134     maxoccupants=u2(1);
135     minresponse=u3(1);
136     outdoortemp=u4(1);
137 else
138     % set parameters depending on time of day
139     if (cn≥'00:00:00')&&(cn<'09:00:00')
140         minoccupant=u1(1);
141         maxoccupants=u2(1);
142         minresponse=u3(1);
143         outdoortemp=u4(1);
144     elseif (cn≥'09:00:00')&&(cn<'11:00:00')
145         minoccupant=u1(2);
146         maxoccupants=u2(2);
147         minresponse=u3(2);
148         outdoortemp=u4(2);
149     elseif (cn≥'11:00:00')&&(cn<'13:00:00')
150         minoccupant=u1(3);
151         maxoccupants=u2(3);
152         minresponse=u3(3);
153         outdoortemp=u4(3);
154     elseif (cn≥'13:00:00')&&(cn<'15:00:00')
155         minoccupant=u1(4);
156         maxoccupants=u2(4);
157         minresponse=u3(4);
158         outdoortemp=u4(4);
```

```
159         elseif (cn≥'15:00:00')&&(cn<'17:00:00')
160             minoccupant=u1(5);
161             maxoccupants=u2(5);
162             minresponse=u3(5);
163             outdoortemp=u4(5);
164         elseif (cn≥'17:00:00')&&(cn<'19:00:00')
165             minoccupant=u1(6);
166             maxoccupants=u2(6);
167             minresponse=u3(6);
168             outdoortemp=u4(3);
169         elseif (cn≥'19:00:00')&&(cn≤'23:59:59')
170             minoccupant=u1(7);
171             maxoccupants=u2(7);
172             minresponse=u3(7);
173             outdoortemp=u4(2);
174         end
175     end
176
177     % force minimum number of responses to minimum number of occupants if
178     % minimum number of reponses is greater than minimum number of occupants
179     if (minresponse>minoccupant)
180         minresponse=minoccupant;
181     end
182
183     % set number of occupants to be within or equal to lower and upper
184     % threshold vslues
185     occupants=randi([minoccupant maxoccupants]);
186
187     % set number of returned responses between minimum number of expected
188     % responses and less than or equal to the total number of occupants
189     response=randi([minresponse occupants]);
190
191     % compute percentage of returned responses; if zero occupants set
192     % percentage to zero
193     if (occupants>0)
194         percent=ceil(response/occupants*100);
195     else
196         percent=0;
197     end
198
199     % set outdoor temperature bias array. Length of array is fixed to total
200     % number of responses. Each array element is in a set defined by vector
201     % (v1, v2 or v3). The range of each vector is defined.
202     % Example >> set=v1(randi(3,response,1))
203     % For each response set the variable 'set' with a randomly selected
204     % element from array v1.
205     switch outdoortemp
206         case 1
207             set=v1(randi(3,response,1));
208         case 2
209             set=v2(randi(4,response,1));
210         case 3
211             set=v2(randi(3,response,1));
212         case 4
213             set=v3(randi(4,response,1));
214         case 5
215             set=v3(randi(3,response,1));
216     end
217
218     %% Compute Comfort Value
219
220     % compute average (mean, mode or median) if total number of responses has
221     % exceeded threshold (nominally minimum of 40%). If threshold condition has
222     % not been satisfied force average to zero.
223
224     if (percent==100)&&(occupants==1)
225         minth=char([89 101 115]);
226         calc_mean=set(1);
227         calc_mode=set(1);
228         calc_med=set(1);
229     elseif(percent≥responsethreshold)
230         minth=char([89 101 115]);
231         calc_mean=mean(set);
232         calc_mode=mode(set);
233         calc_med=median(set);
234     else
235         calc_mean=0;
236         calc_mode=0;
237         calc_med=0;
238         minth=char([78 111]);
239     end
240
```

```
241 %% Comfort Information
242
243 % display to command window comfort information
244  if (info)
245
246      % CREATE STRUCT FOR COMFORT INFO
247      S=struct('maxoccupants',maxoccupants,...
248           'occupants',occupants,...
249           'response',response,...
250           'response_percent',percent,...
251           'response_threshold',responsethreshold,...
252           'threshold_exceeded',minth,...
253           'outdoor_temp',outdoortemp,...
254           'cn',cn,...
255           'HN',sum(set==-2),...
256           'N',sum(set==-1),...
257           'Z',sum(set==0),...
258           'P',sum(set==1),...
259           'HP',sum(set==2),...
260           'calc_mean',mean(set),...
261           'calc_median',median(set),...
262           'calc_mode',calc_mode);
263
264      disp(S);
265      assignin('base','comfort_info',S);
266  end
267
268 %% Assign Variables to the Workspace
269  cv(:,1)=occupants;
270  cv(:,2)=response;
271  cv(:,3)=calc_mode;
272
273  end
```

Listing D.1 comfort_2.m

D.7 date2sec.m

```
function [YVEC] = date2sec(Y)
%%DATE2SEC Rounds date and time to nearest second
%
%  [YVEC] = date2sec(Y) rounds date and time to nearest second. The
%  output Y is serial date number.

%% MATLAB Function Description
%
%  Title: date2sec
%  Filename: date2sec.m
%  Prepared by: Sean Williams
%  Date: 19 Dec 2019
%
%  MATLAB function rounds date time to nearest second
%

%% Change History
%
%   1. [19-12-2019] Initial
%

YVEC=datenum(dateshift(datetime(datestr(Y)),'start','second','nearest'));
```

Listing D.2 date2vec.m

D.8 date2vec.m

```
function [YVEC] = date2vec(Y)
%%DATE2VEC Converts date and time to vector components
%
%  [YVEC] = date2vec(Y) converts date and time to vector components. The
%  output represents the date and time components of hours, minutes,
%  seconds, day, month and year. Y is serial date number.

%% MATLAB Function Description
%
%  Title: date2vec
%  Filename: date2vec.m
%  Prepared by: Sean Williams
%  Date: 1 Aug 2019
%
%  MATLAB function converts date and time to vector components
%

%% Change History
%
%    1. [01-08-2019] Initial
%

%% Convert date and time to vector component

 [yy,mm,dd,hh,mn,se]=datevec(Y);
YVEC(1,1)=hh;
YVEC(1,2)=mn;
YVEC(1,3)=se;
YVEC(1,4)=dd;
YVEC(1,5)=mm;
YVEC(1,6)=yy;
```

Listing D.3 date2vec.m

D.9 demand.m

```
function [dv] = demand(dn_date,info)
%DEMAND computes demand value for EOS
%   DEMAND(DATE,INFO[N]) uses cubic spline coefficients to compute demand value
%   based on analysis of chronological sequence of discrete observations.
%   Polynomial coefficient structure in computed directly from observations
%   after piecewise aggregated approximation has been applied. Separate
%   structures for week day and weekend day are prepared. Monthly
%   variations (seasonal adjustments) are applied using PAA Look Up Table
%   (LUT). The demand value returned has been optimised (rescaled) for use
%   with an Energy Optimisation System (EOS). The option to display demand
%   information (including plot) is set by logic operator INFO [TRUE]/False
%   for example the following instruction computes the demand value at
%   Wed 6th March 2019 at 07:00:00, demand info is selected OFF by default.
%
%   Example:
%   demand=demand('06-Mar-2019 07:00:00',1)
%
%                    current time: Wed 06-Mar-2019 07:00
%                     horizon_hrs: 4
%                              ai: 98.0220
%                              bi: -0.5180
%                              ci: -0.2970
%                              di: 0.0440
%                        time_idx: 22
%                              lo: 22
%                        nd_value: 98.0220
%              nd_value_month_ndm: 111.1464
%           nd_value_month_rescale: 0.9102
%              Euclidean_Distance: 0.0401
%                            RMSE: 4.0076
%                             MAE: 2.8321
%
%   demand=0.9102
%
%   For loop calculation using dijkstra algorithm use:
%       S0_date=now;
%       n=0; % sets multiple of 10 minutes ahead
%       Sn_date=datetime(S0_date,'ConvertFrom','datenum')+minutes(10*n);
%       [dv]=demand(Sn_date) % display demand info and chart optional
%

% Check number of inputs.
if nargin>2
    error('myfuns:nd:TooManyInputs', ...
        'requires at most 2 inputs');
end

% Fill in unset optional values.
switch nargin
    case 1
        info=0;
end

%% MATLAB Function Description
%
%  Title: National Demand
%  Filename: demand.m
%  Prepared by: Sean Williams
%  Date: 17-07-2019
%  MATLAB function computes national demand value with monthly variation
%  for Energy Optimisation Solution (EOS) algorithm
%  Perquisite: demand_initialise.mat, demand_info.mat

%% Change History
%
% # [17-07-2019] Initial
% # [23-07-2019] Bug fixes: correct month identified at end of month
% calcs. plus improvements to plot if selected to display demand info.
% # [25-07-2019] Compute Euclidean Distance (doubled-scaled), RMSE and
%   MAE and include in struct demand_info.
% # [08-08-2019] Introduce new set of spline coefs for concurrent days
%   when calling fcn from Simulink model. dn_date format compatible with
%   Simulink model (Serial Date Number). Removed |mtwtfPAAval| and |ssPAAval|
%   from |nd_demand_info_data.mat| ; these have been replaced with updated
%   array that suports concurrent days. Introduced two external data files:
%   Compute parameters |demand_initialise.mat| and Create plot
```

```
77  %   |demand_info.mat|.
78  % # [27-08-2019] Set demand_initialise.mat variable horizon=0 (previously
79  % 4). Updated description to detail format of function to calculate time
80  % now plus 10 minutes ahead.
81  % # [11-01-2020] Option to change performance evaluation indices to
82  % calculate Euclidean, RMSE and MAE for specific date time (set in
83  % dn_date). See note a L222.
84
85  %% Initialise Variables
86
87  % set start date time variable
88  dn_date = datetime(dn_date,'ConvertFrom','datenum');
89
90  % load data required to compute parameters, including:
91  %   > cubic spline coefs
92  %   > horizon
93  %   > monthLUT
94  data=load('demand_initialise.mat');
95
96  %% Compute Demand Value
97
98  % format date time
99  current_time=datetime((dn_date),'Format','eee dd-MMM-yyyy HH:mm:ss');
100
101 % set date time plus horizon
102 horizon_time=current_time+duration(data.horizon,0,0);
103
104 % find Monday prior to start date
105 m1=dateshift(horizon_time,'dayofweek','Monday','previous');
106
107 % find hour & minute elements of time, choose to ignore seconds
108 [hh,mm,~]=hms(horizon_time-dateshift(horizon_time,'start','day'));
109
110 % compute time index based on current hour. The time index is used to set
111 % the coefs_idx (row), i.e. which set of polynominal coefs (ai bi ci di) to
112 % use to calculate the demand value.
113 time_idx=(hh*2)+(2*mm/60);
114
115 % compute lo value required for polynomial
116 % s(x)=ai+bi(x-lo)+ci(x-lo)^2+di(x-lo)^3; where x=nd_value
117 if hh==0
118     lo=0;
119 else
120     lo=2*((hh-fix(hh/2)-1)+hh-fix(hh/2));
121 end
122
123 % compute s(x) coefficient, i.e. which row of coeff matrix
124 if (hh>=0) && (hh<1)
125     coefs=1;
126 elseif (hh>=1)&&(hh<23)
127     coefs=hh-fix(hh/2)+1;
128 elseif (hh>=23)
129     coefs=hh-fix(hh/2)+1;
130 end
131
132 % calculate number of days from Monday to start date
133 Δday=caldays(between(m1,horizon_time,'days'))+1;
134
135 % calculate month number (April=1)
136 Δmonth=mod(calmonths(between('1-April-2005',horizon_time)),12)+1;
137
138 % set cublic spline interpolation polynomial coefficients
139 switch (Δday)
140     case num2cell(1:4) % cubic spline polynomial coefs weekday: MTWT
141         cscoefs=data.coeff(:,:,1);
142         cscoefs(14:16,:)=data.coeff(1:3,:,1);
143         type=1;
144     case 5 % cubic spline polynomial coefs weekday: F
145         cscoefs=data.coeff(:,:,1);
146         cscoefs(14:16,:)=data.coeff(1:3,:,2);
147         type=1;
148     case 6 % cubic spline polynomial coefs weekend day: Sa
149         cscoefs=data.coeff(:,:,2);
150         cscoefs(14:16,:)=data.coeff(1:3,:,2);
151         type=2;
152     case 7 % cubic spline polynomial coefs weekend day: Su
153         cscoefs=data.coeff(:,:,2);
154         cscoefs(14:16,:)=data.coeff(1:3,:,1);
155         type=2;
156 end
157
158 % assign coefficent values
```

```
159  ai=cscoefs(coefs,1);
160  bi=cscoefs(coefs,2);
161  ci=cscoefs(coefs,3);
162  di=cscoefs(coefs,4);
163
164 % compute demand at current time plus horizon (excluding Month LUT)
165  nd_value=ai+bi*(time_idx-lo)+ci*(time_idx-lo)^2+di*(time_idx-lo)^3;
166
167 % compute national demand at current time plus horizon (inc. Month LUT)
168  nd_value_month=nd_value+data.monthLUT(Δmonth)-data.monthLUT(1);
169
170 % resample nd_value_month
171  nd_value_month_rescale=rescale(nd_value_month,0,1,'InputMin',...
172      data.cs365_min,'InputMax',data.cs365_max);
173
174 %% Demand Information
175
176 % display to command window comfort information and plot output
177  if (info)
178      % load parameters required to create plot
179       info=load('demand_info.mat');
180
181      X4=current_time+duration(data.horizon,0,0);
182      D4=duration(str2double(strsplit(datestr(X4,'HH:MM:ss'),':')));
183
184      switch type
185          case 1
186              PAA_data=info.mtwtfPAA_data;
187              C_data=info.mtwtfdata;
188              chart_label='Weekday';
189              pos='southeast';
190          case 2
191              PAA_data=info.ssPAA_data;
192              C_data=info.ssdata;
193              chart_label='Weekend day';
194              pos='northeast';
195      end
196
197      % CLAMPED SPLINE INTERPOLATION PLOT
198      y=cat(1,PAA_data(1,1),PAA_data);
199      y=cat(1,y,PAA_data(end,1));
200      qy=spline(info.t4,[0 y' 0]);
201      qx=linspace(0,47,48);
202      qe=ppval(qy,qx);
203      qe=qe+data.monthLUT(Δmonth)-data.monthLUT(1);
204
205      % COMPUTE DOUBLE-SCALED EUCLIDEAN DISTANCE
206      % scaling it into a range defined by 0 through to maximum possible
207      % distance obseravle between the two variables:
208      % qe=cubic spline interpolation
209      % C_data=cumulative mean data (either weekday or weekend day)
210
211      % determine the maximum possible squared discrepancy for each variable
212      % comparison using the mdi_min and mdi_max values:
213      mdi_min=0;
214      mdi_max=100;
215      mdi=(mdi_max-mdi_min)^2;
216
217      % compute the sum of squared discrepancies per variable, dividing
218      % through the squared discrepancy for eah variable by the maximum
219      % possible discrepancy for that variable. Then take the square root of
220      % the sum to produce the scaled variable Euclidean distance:
221
222      % Note: option to calculate performance at specific point (that is as
223      % defined by date time stamp) rather than for whole plot data
224      % (default). To set for specific date time insert <time_idx> as index
225      % refernce from parameters: qe and C_data. Also need to change divide by
226      % value to just 1. Same applies for rmse and mae.
227      d1=sum(((qe-(C_data+data.monthLUT(Δmonth)-data.monthLUT(1))).^2)/mdi);
228      % compute the scaled value by dividing by the square root of the number
229      % of variables:
230      d2=sqrt(d1)/(sqrt(size(qe,2)));
231
232      % COMPUTE RMSE
233      rmse=sqrt(sum((qe-(C_data+data.monthLUT(Δmonth)-data.monthLUT(1))).^2)/(size(qe,2)));
234
235      % COMPUTE MAE
236      mae=sum(abs(qe-(C_data+data.monthLUT(Δmonth)-data.monthLUT(1))))/(size(qe,2));
237
238      % CREATE NEW FIGURE
239       figure('Name','Demand Data Info','NumberTitle','off');
240      % p1=plot clamped spline interpolation based on piecewise polynomial qe - color blue
```

```
241       p1=plot(info.D48,qe,'b-','LineWidth',1.2);
242       hold on
243       p2=plot(info.D48,C_data+data.monthLUT(Δmonth)-data.monthLUT(1),'m--');
244       % create marker showing demand value (month) and placeholder required
245       % for legend X, Y and d (demand value month rescaled).
246       p3=plot(D4,nd_value_month,'wo','LineWidth',1.5,'MarkerSize',9,'MarkerFaceColor','w');
247       p4=plot(D4,nd_value_month,'wo','LineWidth',1.5,'MarkerSize',9,'MarkerFaceColor','w');
248       p5=plot(D4,nd_value_month,'ro','LineWidth',1.5,'MarkerSize',9);%,'MarkerFaceColor','w');
249
250       hold off
251
252       % chart format
253       lgd={'Spline-coefs','Cumulative Mean',['t: ',char(D4,'hh:mm'),...
254            ', ndm: ',num2str(nd_value_month,'% .2f')],...
255            ['Rescale (\bf',num2str(nd_value_month_rescale,'% .3f'),'\rm)'],...
256            ['Euclidean: ', num2str(d2,'%0.4f')]};
257       lgd=legend([p1 p2 p5 p3 p4],lgd,'Location',pos);
258
259       title(lgd,['Predicted Info (', char(datetime(horizon_time,'Format','dd-MMM-yyyy')),')']);
260       title({[char(chart_label), ' Demand Profile'];['Current time: ' char(current_time)]});
261       xlabel('Time (HH:MM)');
262       ylabel('Demand (Rescaled)');
263       grid on
264       axis tight
265       ylim([-15 135])
266       xtickformat('hh:mm');
267
268       % CREATE STRUCT FOR DEMAND INFO
269       S=struct('current_time',{current_time},...
270                'horizon_hrs',{data.horizon},...
271                'ai',{ai},...
272                'bi',{bi},...
273                'ci',{ci},...
274                'di',{di},...
275                'time_idx',{time_idx},...
276                'lo',{lo},...
277                'nd_value',{nd_value},...
278                'nd_value_month_ndm',{nd_value_month},...
279                'nd_value_month_rescale',{nd_value_month_rescale},...
280                'Euclidean_Distance',{d2},...
281                'RMSE',{rmse},...
282                'MAE',{mae});
283
284       disp(S)
285
286       assignin('base','demand_info',S);
287       assignin('base','qe',qe);
288
289   end
290
291   %% Assign variables to the workspace
292
293   dv=nd_value_month_rescale;
294
295   end
```

Listing D.4 demand.m

D.10 demo_dtv.m

```
function [measured_temp] = demo_dtv(data)
%%DEMO_DTV Option to use measured outdoor tempaerature variation
%
%   [MEASURED_TEMP] = demo_dtv(data) returns measured temperature
%   variation in degC for selected date.
%   Where DATA(1,1)=date_time_option indicator, DATA(2,1)=count increments every
%   Timer_int (sec) [1800] and DATA(2:49,1)=measured outdoor temperature variation
%   for selected date_time (set in [optim_ctrl_model_data.m]

%% MATLAB Function Description
%
%  Title: Demonstration Daily Temperature Variation
%  Filename: demo_dtv.m
%  Prepared by: Sean Williams
%  Date: 24 Dec 2019
%
%  MATLAB function returns measured temperature variation in degC
%

%% Change History
%
%   1. [24-18-2019] Initial

%% Calculate Measured Temperature

date_time_option=data(1,1);
count=data(2,1);
daily_temp=data(3:end,1);

if (ge(date_time_option,3))
    %daily_temp=data(3:50,1);
    % convert measured data from degF and degC
    measured_temp = (daily_temp(count,1)-32)*(5/9);
else
    measured_temp=0;
end
```

Listing D.5 demo_dtv.m

D.11 dijkstra.m

```
function [cost, path] = dijkstra(G,s,t)
%%DIJKSTRA Computes a gridmap cost and shortest path.
%
%  Construction:
%   [COST,PATH] = DIJKSTRA(G,S,T) computes the cost and shortest path
%   starting at node S and ending at node T. Where G is a weighted graph
%   (digraph) that has been converted initially into a sparse adjacency
%   matrix then into a full matrix. The PATH contains all the nodes on the
%   shortest path. The weights between each node on the shortest path are
%   returned as a single value representing the total cost from node S to
%   node T.
%

%% Additional Information
%
%  Algorithm constructs a weighted matrix table. The start node is stored
%  in col 1 index 0 with remaining nodes in col 2 to n; where n is
%  number of nodes.
%
%  adjacency matrix
%  G=[0 5 10 0 0 0 0;
%     0 0 0  6 3 0 0;
%     0 0 0  0 0 0 0;
%     0 0 0  0 0 6 0;
%     0 0 2  2 0 0 2;
%     0 0 0  0 0 0 0;
%     0 0 0  0 0 2 0];
%
%  [cost, path]=dijkstra(G,1,6)
%    cost = 12
%    path = 1 2 5 7 6
%
%  weighted matrix table
%   from | to...
%    0(A) 2(B) 3(C) 4(D) 5(E) 6(F) 7(G)
%    0(A)  5    10   inf  inf  inf  inf
%    2(B)  5    10   11   8    inf  inf
%    5(E)  5    10   10   8    inf  10
%    3(C)  5    10   10   8    inf  10
%    4(D)  5    10   10   8    16   10
%    7(G)  5    10   10   8    12   10
%
%  path A(1) to F(6) is found in reverse order: F<G<E<B<A which translates
%  to A(1)>B(2)>E(5)>G(7)>F(6)
%

%% MATLAB Function Description
%
%  Title: Dijkstra's Shortestpath Algorithm
%  Filename: dijkstra.m
%  Prepared by: Sean Williams
%  Date: 19-06-2019
%
%  MATLAB function computes cost and shortestest path between nominated
%  start (S) and end (T) nodes.
%

%% Change History
%
% # [19-06-2019] Initial.
%

%% Dijkstra's Algorithm

% check start and end nodes
 if s==t
     cost=0;
     path=s;
 else
     % initialise adjacency matrix replace 0 with inf
     for i=1:size(G,1)
         for j=1:size(G,1)
             if G(i,j)==0
                 G(i,j)=inf;
             end
         end
```

```
77      end
78
79      if t==1
80          t=s;
81      end
82
83      % exchange row s for row 1
84      temp=G(:,1);
85      G(:,1)=G(:,s);
86      G(:,s)=temp;
87
88      % exchange col s for col 1
89      temp=G(1,:);
90      G(1,:)=G(s,:);
91      G(s,:)=temp;
92      lenG=size(G,1);
93
94      % initialise weight matrix table (wmt)
95      wmt=zeros(lenG);
96
97      % populate wmt
98      for i=2:lenG
99          wmt(1,i)=i;
100         wmt(2,i)=G(1,i);
101     end
102
103     % set col 1 as row 1, then col 2 1:1:lenG
104     node_1=zeros(lenG,2);
105     for i=1:lenG
106         node_1(i,1)=G(1,i);
107         node_1(i,2)=i;
108     end
109
110     node_2=node_1(2:length(node_1),:);
111     path=2;
112
113     % sort wmt
114     while le(path,(size(wmt,1)-1))
115         path=path+1;
116         node_2=sortrows(node_2,1);
117         k=node_2(1,2);
118         wmt(path,1)=k;
119         node_2(1,:)=[];
120         for i=1:size(node_2,1)
121             if gt(node_1(node_2(i,2),1),(node_1(k,1)+G(k,node_2(i,2))))
122                 node_1(node_2(i,2),1) = node_1(k,1)+G(k,node_2(i,2));
123                 node_2(i,1) = node_1(node_2(i,2),1);
124             end
125         end
126         for i=2:length(G)
127             wmt(path,i)=node_1(i,1);
128         end
129     end
130
131     if t==s
132         path=1;
133     else
134         path=t;
135     end
136
137     % find cost from start to end nodes from wmt
138     cost=wmt(size(wmt,1),t);
139
140     % format dijsktra shortest path (dsp) to read left to right
141     path=flip(dsp(path,wmt,s,t));
142
143     % assign completed wmt to cell in base
144     assignin('base','wmt',{wmt})
145 end
146
147 function path = dsp(path,wmt,s,t)
148 % compute path from weight matrix table (or display only)
149
150 idx=size(wmt,1);
151 while gt(idx,0)
152     if wmt(2,t)==wmt(size(wmt,1),t)
153         path=[path s];
154         idx=0;
155     else
156         idx2=size(wmt,1);
157         while gt(idx2,0)
158             if lt(wmt(idx2,t),wmt(idx2-1,t))
```

```
159                     path=[path wmt(idx2,1)];
160                     path=dsp(path,wmt,s,wmt(idx2,1));
161                     idx2=0;
162                 else
163                     idx2=idx2-1;
164                 end
165                 idx=0;
166             end
167         end
168     end
```

Listing D.6 dijkstra.m

D.12 initialise.m

```
function [visual_mode,horizon,gridmap,edgepath,t1,temp_step,event_duration] = initialise(S0_date)
%% MATLAB Function Description
%
%  Title: Initialise Parameters
%  Filename: initialise.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function intialises parameters converts date and time to vector components
%

%% Change History
%
%   1. [06-11-2019] Initial
%   2. [27-02-2020] New parameter: visual_mode
%   3. [21-02-2020] Change variable names (interval>temp_step)
%

%% Set Model Parameters

% Visual Mode
% option to display different combinations of gridmap
% visual_mode: [1]=none*
%              [2]=optim
%              [3]=demand,optim
%              [4]=comfort,demand,tou,optim
visual_mode=1;

% Time date variable
t1=datetime(datestr(S0_date));

% Horizon window
% option to plot tou data to 4h or 24h
% compute gridmap (shortest path) fixed to 4h only
horizon=4;

if horizon==4
    horizon=25;
else
    horizon=144;
end

% Temperature path: [2]=initial temperature + 2degC
%                   [3]=initial temperature + 3degC
temp_step=3;

% Demand event duration: [1]=30min
%                        [2]=40min
%                        [3]=50min
event_duration=2;

% load gridmap template from compresseed MAT-file; contains both 1-D and
% multidimensional array
load('grid4_1.mat','grid4_0');

% Gridmap to multidimensional array template (31x72x4 double)
gridmap=grid4_0;

% clear edgepath
edgepath=[]; % clear edgepath

% end initialise
```

Listing D.7 initialise.m

D.13 optim_ctrl.m

```
function optim_ctrl(block)
%% OPTIM_CTRL
%
%  Title: Optimisation and Control
%  Filename: optim_ctrl.m
%  Prepared by: Sean Williams
%  Date: 20 Dec 2019
%
%  Code tagged to Simulink model block optimise_subsystem. On receipt of
%  date/time (sampe rate: 10 minute) code computes new temperature
%  setpoint (ctrl_action) using Dijsktra's algorithm which is a function
%  of occupant thermal comfort, electricity demand and cost (tariff).
%  Code reacts on receipt of demand event signal
%

setup(block);

%endfunction: optim_ctrl(block)

function setup(block)
%% Setup Functional Port Properties

% Register number of ports
block.NumInputPorts  = 3;
block.NumOutputPorts = 3;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Override input port properties
block.InputPort(1).Dimensions  = 1;  % temp_room
block.InputPort(1).DatatypeID  = 0;  % double
block.InputPort(1).Complexity  = 'Real';
block.InputPort(1).DirectFeedthrough = true;

% Override input port properties
block.InputPort(2).Dimensions  = 1;  % S0_date
block.InputPort(2).DatatypeID  = 0;  % double
block.InputPort(2).Complexity  = 'Real';
block.InputPort(2).DirectFeedthrough = true;

% Override input port properties
block.InputPort(3).Dimensions  = 1;  % des_mode
block.InputPort(3).DatatypeID  = 0;  % double
block.InputPort(3).Complexity  = 'Real';
block.InputPort(3).DirectFeedthrough = true;


% Override output port properties
block.OutputPort(1).Dimensions  = 1; % ctrl_action
block.OutputPort(1).DatatypeID  = 0; % double
block.OutputPort(1).Complexity  = 'Real';

% Override output port properties
block.OutputPort(2).Dimensions  = 1; % tou_tariff
block.OutputPort(2).DatatypeID  = 0; % double
block.OutputPort(2).Complexity  = 'Real';

% Override output port properties
block.OutputPort(3).Dimensions  = 1; % des_duration
block.OutputPort(3).DatatypeID  = 0; % double
block.OutputPort(3).Complexity  = 'Real';

% Register parameters
block.NumDialogPrms = 0;

% Register sample times
block.SampleTimes = [600  0];

% Specify the block simStateCompliance to default
block.SimStateCompliance = 'DefaultSimState';

% Register nethods
block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
```

```
77  block.RegBlockMethod('Start', @Start);
78  block.RegBlockMethod('Outputs', @Outputs); % Required
79  block.RegBlockMethod('Update', @Update);
80  block.RegBlockMethod('Derivatives', @Derivatives);
81  block.RegBlockMethod('Terminate', @Terminate); % Required
82  block.RegBlockMethod('SetInputPortSamplingMode',@SetInpPortFrameData);
83
84  %endfunction: setup(block)
85
86  function DoPostPropSetup(block)
87
88  % Initialise the Dwork vectors
89  block.NumDworks = 4;
90
91  % Dwork(1) stores the status of the count_flag [count_flag]
92  block.Dwork(1).Name            = 'D1';
93  block.Dwork(1).Dimensions      = 1;
94  block.Dwork(1).DatatypeID      = 0;      % double
95  block.Dwork(1).Complexity      = 'Real'; % real
96  block.Dwork(1).UsedAsDiscState = true;
97
98  % Dwork(2) stores the value of the counter [count]
99  block.Dwork(2).Name            = 'D2';
100 block.Dwork(2).Dimensions      = 1;
101 block.Dwork(2).DatatypeID      = 0;      % double
102 block.Dwork(2).Complexity      = 'Real'; % real
103 block.Dwork(2).UsedAsDiscState = true;
104
105 % Dwork(3) stores the nodepath as a vector when a dv event is initiated [dv_event]
106 block.Dwork(3).Name            = 'D3';
107 block.Dwork(3).Dimensions      = 25;
108 block.Dwork(3).DatatypeID      = 0;      % double
109 block.Dwork(3).Complexity      = 'Real'; % real
110 block.Dwork(3).UsedAsDiscState = true;
111
112 % Dwork(4) stores the status of the des_end [des_end]
113 block.Dwork(4).Name            = 'D4'; %des_end
114 block.Dwork(4).Dimensions      = 1;
115 block.Dwork(4).DatatypeID      = 0;      % double
116 block.Dwork(4).Complexity      = 'Real'; % real
117 block.Dwork(4).UsedAsDiscState = true;
118
119 %endfunction: DoPostPropSetup(block)
120
121 function InitializeConditions(block)
122 %% Set Initial Conditions
123
124 % Set the initial status of the count_flag to zero
125 block.Dwork(1).Data=1;
126
127 % Set the initial value of demand event counter to 24
128 % 24 is baseline counter required for 4 hour ramp time, and before event
129 % duration counter is applied
130 block.Dwork(2).Data=24;
131
132 % Set initial status of des_end to zero
133 % des_end=0 [no demand event signal]
134 block.Dwork(4).Data=0;
135
136 % Set Simulink Model block parameters
137 set_param('optim_ctrl_model_sim/des_subsystem/des_end','Value','0')
138
139 % Set des_duration output signal to zero
140 block.OutputPort(3).Data=0;
141
142 %endfunction: InitializeConditions(block)
143
144 function Start(block)
145 %% Set Start Conditions
146
147 % Assign Dwork(1) to status of count_flag
148 block.Dwork(1).Data=1;
149
150 % Assign Dwork(4) status of des_end
151 block.Dwork(4).Data=block.InputPort(3).Data;
152
153 %endfunction: Start(block)
154
155 function Outputs(block)
156 %% Outputs
157
158 % Set Simulink blck parameters
```

```
159  temperature=block.InputPort(1).Data;
160  des_end=block.Dwork(4).Data;
161  count_flag=block.Dwork(1).Data;
162  count=block.Dwork(2).Data;
163  dv_event=block.Dwork(3).Data;
164  set_param('optim_ctrl_model_sim/des_subsystem/des_end',...
165      'Value','0'); % reset des_end at end of event
166
167 % set S0_date to 'base' for initial cycle only, then revert to date on Input Port 4
168  if block.InputPort(2).Data<10
169      S0_date=datetime(datestr(evalin('base','dt')));
170  else
171      S0_date=datetime(datestr(block.InputPort(2).Data));
172  end
173
174 % Diagnostic view [S0_date]
175 %S0_date
176
177 % Initialise model
178  [visual_mode,horizon,gridmap,edgepath,t1,temp_step,event_duration]=initialise(S0_date);
179
180 % Set edgepath for each function
181 %       [1] comfort
182 %       [2] demand
183 %       [3] tou (tariff)
184 %       [4] optim (ALL)
185  for edgepage=1:4
186
187      switch edgepage
188
189          case 1 % comfort
190
191              % 1. INITIALISE (LOCAL)
192              % =====================
193              % set local paramaters
194
195              % set minimum temperature threshold
196              % if nil occupancy edge weight will force path to reduce temperature
197              % setpoint until minimum temperature threshold is reached. At which point
198              % edge weight will force maintain minimum temperature threshold
199              mintempthreshold=16;
200
201              % set minimum temperature threshold parameter
202              % translate minimum temperature threshold to equivalent grid map value
203              mintemp=[17.5 17 16.5 16 15.5];
204              mintempvar=[17 20 23 26 29];
205              mintempthresholdparam=mintempvar(mintemp==mintempthreshold);
206
207              % define array that describes range of temperatures (nodes) at time t0
208              t0tempSP=[20.5 20 19.5 19 18.5 18 17.5 17 16.5 16 15.5];
209              % t0nodes=1:1:11;
210
211              % 2a. PREPARE COMFORT VALUES
212              % ==========================
213              [tempSP]=prepare_comfort_values(temperature);
214
215              % 3. PREPARE GRIDMAP
216              % ==================
217              [tcv,tcvdata,gridmap_c,t0minidx,t240minidx]=prepare_tc_gridmap(t0tempSP,tempSP,...
218                  S0_date,gridmap,edgepath,edgepage,mintempthresholdparam);
219
220              % 4. PREPARE DIGRAPH
221              % ==================
222              [G_c,B]=prepare_digraph(gridmap_c,edgepage);
223
224              % 5. DIJKSTRA
225              % ===========
226              [¬,path_c]=dijkstra(B,gridmap_c(t0minidx,1,edgepage),gridmap_c(t240minidx,71,...
227                  edgepage));
228              [edgepath_c]=prepare_edgepath(gridmap_c,edgepage,path_c);
229
230              % 6. VISUALISATION: COMFORT (4H/24H HORIZON WINDOW)
231              % ================================================
232              % fixed subplot showing scrolling comfort [1,4,1]
233              if (visual_mode==4)
234                  visual_comfort_data(horizon,tcvdata,t1);
235              end
236
237              % 7. VISUALISATION: GRIDMAP INDIVIDUAL SHORTESTPATH
238              % =================================================
239              % new figure created for each cycle showing individual gridmap
```

```
240             %fig_name='Comfort Gridmap';
241             %edgepath_color=[153/255 0 0]; % red
242             %visual_individual_shortestpath(fig_name,G_c,edgepage,edgepath_c,t1,edgepath_color);
243
244         case 2 % demand
245
246             % 1. INITIALISE (LOCAL)
247             % =====================
248
249
250             % 2a. PREPARE DEMAND VALUES
251             % =========================
252             [dv,dv_gridmap,dv_nodepath,dv_rescale]=prepare_dv_values(horizon,S0_date);
253
254             % 2b. PREPARE NODE PATH
255             % =====================
256                 if(des_end==1)
257                     % on receipt of demand event signal set grid path to increase by 2degC or
258                     % 3degC [temp_step] from last recorded temperature over 4hr horizon window at
259                     % increments of 0.5degC every 50 min [2degC] or 40 min [3degC].
260                     if (count_flag==1)
261                         block.OutputPort(3).Data=0;
262
263                         count_flag=0; % set flag to false to ensure loop is executed only once
264                         block.Dwork(1).Data=count_flag;
265                         % set node start point (source) temperature
266                         node=dv_rescale(1,:);
267                         %node=10 % test
268
269                         dv_nodepath=[]; % temporary placeholder for new path
270
271                         switch temp_step
272                             case 2
273                             dv_nodepath=zeros(25,1); % initialise variable dv_nodepath
274                             % set path for 2degC increase over 4hr period increasing 0.5degC at
275                             % 50min intervals
276                                 for p1=0:5:20
277                                     for p2=1:5
278                                         if node-(p1/5)<1
279                                             dv_nodepath(p2+p1,1)=1;
280                                         else
281                                             dv_nodepath(p2+p1,1)=node-(p1/5);
282                                         end
283                                     end
284                                 end
285                             case 3
286                             dv_nodepath=zeros(25,1); % initialise variable dv_nodepath
287                             % set path for 3degC increase over 4hr period increasing 0.5degC at
288                             % approximately 40min intervals
289                                 for p1=0:4:25
290                                     for p2=1:4
291                                         if node-(p1/4)<1
292                                             dv_nodepath(p2+p1,1)=1;
293                                         else
294                                             dv_nodepath(p2+p1,1)=node-(p1/4);
295                                         end
296                                     end
297                                 end
298                                 % trim dv_nodepath
299                                 dv_nodepath(1,:)=[];
300                                 dv_nodepath(26:end,:)=[];
301                         end
302                         dv_event=dv_nodepath;
303                         block.Dwork(3).Data=dv_event;
304
305                     % on receipt of demand event signal, and after inital path showing
306                     % increase of either 2degC or 3degC [temp_step] code begins to scroll
307                     % grid map horizontally by one-step at each 10 minute cycle.
308                     else
309                         if (count>1)
310                             dv_nodepath(1:count-2,1)=dv_event(size(dv_nodepath,1)-count+2: ...
311                                 end-1,:);
312                             if ((dv_event(1,1)+temp_step)>11)
313                                 dv_nodepath(count-1:count+event_duration,:)=11;
314                             else
315                                 dv_nodepath(count-1:count+event_duration,:)=dv_event(1)+2;
316                             end
317                         else
318                             if ((dv_event(1,1)+temp_step)>11)
319                                dv_nodepath(1:count+event_duration,:)=11;
320                             else
321                                dv_nodepath(1:count+event_duration,:)=dv_event(1)+2;
```

```
                        end
                    end
                    count=count-1;
                    block.Dwork(2).Data=count;

                    % this will trigger use of ESS for duration of
                    % demand side event, try this first then remember
                    % to reset new output back to zero in the next if
                    % loop below
                    if count==1
                        block.OutputPort(3).Data=3;
                    end

                    if (count==-event_duration-1)
                        set_param('optim_ctrl_model_sim/des_subsystem/des_end',...
                            'Value','1');
                        count_flag=1;
                        block.Dwork(1).Data=count_flag;
                        count=24; % reset count
                        block.Dwork(2).Data=count;
                        block.Dwork(4).Data=0;
                    end
                end
            end

        % 3. PREPARE GRIDMAP
        % ==================
        [t0minidx,t240minidx,gridmap_d]=prepare_gridmap(dv_nodepath,gridmap,edgepage,...
            edgepath);

        % 4. PREPARE DIGRAPH
        % ==================
        [G_d,B]=prepare_digraph(gridmap_d,edgepage);

        % 5. DIJKSTRA
        % ===========
        [~,path_d]=dijkstra(B,gridmap_d(t0minidx,1,edgepage),gridmap_d(t240minidx,71,...
            edgepage));
        [edgepath_d]=prepare_edgepath(gridmap_d,edgepage,path_d);

        % 6. VISUALISATION: Demand (4H/24H HORIZON WINDOW)
        % ================================================
        % fixed subplot showing scrolling demand [1,4,2]
        if (visual_mode==3) || (visual_mode==4)
            visual_demand_data(horizon,dv,dv_gridmap,dv_rescale,t1)
        end

        % 7. VISUALISATION: GRIDMAP INIDVIDUAL SHORTESTPATH
        % ================================================
        % new figure created for each cycle showing individual gridmap
        %fig_name='Demand Gridmap';
        %edgepath_color=[0 112/255 192/255]; % blue
        %visual_individual_shortestpath(fig_name,G_d,edgepage,edgepath_d,t1,edgepath_color)

    case 3 % tou

        % 1. INITIALISE (LOCAL)
        % =====================
        % set local parameters
        tou_tariff=[0.0499 0.1199 0.2499];
        period={'00:00:00','06:00:00','16:00:00','19:00:00','23:00:00','24:00:00'};

        % 2a. PREPARE TOU VALUES
        % ======================
        [touv_gridmap,touv_nodepath,touv_rescale,touv]=prepare_tou_values(horizon,...
            S0_date,period,tou_tariff);

        touv_op=touv(1,1);

        % 3. PREPARE GRIDMAP
        % ==================
        [t0minidx,t240minidx,gridmap_t]=prepare_gridmap(touv_nodepath,gridmap,edgepage,...
            edgepath);
        %gridmap_t=gridmap;

        % 4. PREPARE DIGRAPH
        % ==================
        [G_t,B]=prepare_digraph(gridmap_t,edgepage);

        % 5. DIJKSTRA
        % ===========
```

```
403             [¬,path_t]=dijkstra(B,gridmap_t(t0minidx,1,edgepage),gridmap_t(t240minidx,71,...
404                 edgepage));
405             [edgepath_t]=prepare_edgepath(gridmap_t,edgepage,path_t);
406 
407             % 6. VISUALISATION: TOU (4H/24H HORIZON WINDOW)
408             % ============================================
409             % fixed subplot showing scrolling tou [1,4,3]
410             if (visual_mode==4)
411                 visual_tou_data(horizon,touv_rescale,t1)
412             end
413 
414             % 7. VISUALISATION: GRIDMAP INDIVIDUAL SHORTESTPATH
415             % ================================================
416             % new figure created for each cycle showing individual gridmap
417             %fig_name='TOU Gridmap';
418             %edgepath_color=[204/255 0 153/255]; % magenta
419             %visual_individual_shortestpath(fig_name,G_t,edgepage,edgepath_t,t1,edgepath_color)
420 
421         case 4 % optim
422 
423             % 1. INITIALISE (LOCAL)
424             % =====================
425             % set local parameters
426             stage_centroid=1;
427             X=zeros(3,2);
428 
429             % 2. PREPARE VALUES
430             % =================
431             % not required
432 
433             % 3. PREPARE GRIDMAP
434             % ==================
435             % set gridmap to multidimensional array template (31x72x4 double)
436             gridmap(:,:,1)=gridmap_c(:,:,1);
437             gridmap(:,:,2)=gridmap_d(:,:,2);
438             gridmap(:,:,3)=gridmap_t(:,:,3);
439             gridmap(:,:,4)=gridmap_c(:,:,4);
440 
441             for s=3:3:72
442                 for p=1:(edgepage-1)
443                     X(p,1)=find(gridmap(:,s,p)==min(gridmap(:,s,p)));
444                     X(p,2)=0;
445                     [¬,c]=kmeans(X,1);
446                     id=floor(c(1));
447                 end
448 
449                 gridmap(id,s,edgepage)=stage_centroid;
450                 for j=id-1:-1:1
451                     gridmap(j,s,edgepage)=stage_centroid+id-j;
452                 end
453                 for j=id+1:1:31
454                     gridmap(j,s,edgepage)=stage_centroid+j-id;
455                 end
456             end
457 
458             t0minidx=find(gridmap(:,3,4)==min(gridmap(:,3,4)));
459             t240minidx=find(gridmap(:,72,4)==min(gridmap(:,72,4)));
460 
461             % 4. PREPARE DIGRAPH
462             % ==================
463             [G_a,B]=prepare_digraph(gridmap,edgepage);
464 
465             % 5. DIJKSTRA
466             % ===========
467             [¬,path_a]=dijkstra(B,gridmap(t0minidx,1,edgepage),gridmap(t240minidx,71,edgepage));
468             [edgepath_a]=prepare_edgepath(gridmap,edgepage,path_a);
469 
470             % 6. VISUALISATION: DATA (4H/24H HORIZON WINDOW)
471             % ============================================
472             % not required
473 
474             % 7. VISUALISATION: GRIDMAP INDIVIDUAL SHORTESTPATH
475             % ================================================
476             % not required
477 
478             % 8. CONTROL ACTION
479             % =================
480             % control action is temperature at t10, S1
481             ctrl_stage=2;
482             ctrl_action=t0tempSP(path_a(ctrl_stage)-(11*(ctrl_stage-1)));
483 
```

```
484             % 9. VISUALISATION: BIG PATH
485             % ==========================
486             %fixed subplot showing scrolling total cost function [1,4,4]
487              if (visual_mode==2) || (visual_mode==3) || (visual_mode==4)
488                  visual_group_path(path_c, path_d, path_t, path_a, t1)
489              end
490
491             % 10. VISUALISATION: BIG GRIDMAP SHORTESTPATH
492             % ===========================================
493             % new figure created for each cycle showing individual gridmap
494             %visual_group_shortestpath(G_c,edgepath_c,G_d,edgepath_d,G_t,edgepath_t,G_a,...
495             %edgepath_a,t1,horizon)
496     end
497 end
498
499 % Update Simulink model output ports
500 block.OutputPort(1).Data = ctrl_action ;
501 block.OutputPort(2).Data = touv_op;
502
503
504 %endfunction: Outputs(block)
505
506 function Update(block)
507 %% Update Dwork
508
509 % Update Dwork(4) to InputPort(3) [S0_date]
510 block.Dwork(4).Data=block.InputPort(3).Data;
511
512 %endfunction: Update(block)
513
514 function SetInpPortFrameData(block, idx, fd)
515
516 % Set the sampling of the input ports
517 block.InputPort(idx).SamplingMode=fd;
518 for i=1:block.NumOutputPorts
519     block.OutputPort(i).SamplingMode=fd;
520 end
521
522 %endfunction: SetInpPortFrameData(block,idx,fd)
523
524 function Terminate(block)
525
526 %endfunction: Terminate(block)
```

Listing D.8 optim_ctrl.m

D.14 optim_ctrl_model_data.m

```
%% MATLAB M-File Description
%
%  Title: Optimisation and Control Model Building Parameters
%  Filename: optim_ctrl_model_data.m
%  Prepared by: Sean Williams
%  Date: 24 Oct 2019
%
%  MATLAB script tagged to Simulink model 'optim_ctrl_model_sim.slx',
%

%% Change History
%
%   1. [24-10-2019] Initial
%   2. [20-12-2019] New: Set Initialise Parameters, Set Date Time,
%   (option to use dtv_sim or dtv_act), Set SOC Model Parameters,
%   Set Outdoor Temperature Variation; Modified: Set Building Parameters.
%   3. [26-01-2020] New: energy_subsystem parameters

%% Set Initialise Parameters

Δ1=1.157412771135569e-05;
Δ10=0.006944444445185;

%% Set Date Time

% Option to select simulated daily temperature variation [1|2]
% or measured daily temperature [3|4|5|6]
% https://www.wunderground.com <act_temp.xlsx>

date_time_option=4;

switch date_time_option
    case 1
        date_time='now';
        daily_temp=111;
    case 2
        date_time='10:00:00';
        daily_temp=222;
    case 3 % Sunday 10-Feb-2019 00:00:00, 24hrs
        date_time='10-Feb-2019 00:00:00'; %datenum=737466
        % https://www.wunderground.com/history/daily/gb/newcastle-upon-tyne/EGNT/date/2019-2-10
        daily_temp=[39 37 39 39 37 36 36 36 37 37 37 37 ...
                    36 36 36 36 36 36 37 41 41 43 43 43 ...
                    43 45 43 41 45 45 41 41 39 39 39 37 ...
                    36 36 37 37 36 36 36 37 36 36 36 36 ...
                    36  24  32  34];

    case 4 % Tuesday 07-May-2019 00:00:00, 24hrs
        date_time='7-May-2019 00:00:00'; %datenum=737552
        % https://www.wunderground.com/history/daily/gb/newcastle-upon-tyne/EGNT/date/2019-5-7
        daily_temp=[37 37 37 37 37 37 37 37 37 37 37 37 ...
                    39 39 41 41 43 43 43 45 45 45 45 43 ...
                    45 46 45 46 46 46 46 46 45 45 45 45 ...
                    45 43 43 43 43 43 43 43 43 43  43 ...
                    43  43  43  43];

    case 5 % Saturday 3-Aug-2019 00:00:00, 24hrs
        date_time='3-Aug-2019 00:00:00'; %datenum=737640
        % https://www.wunderground.com/history/daily/gb/newcastle-upon-tyne/EGNT/date/2019-8-3
        daily_temp=[57 55 55 55 55 55 55 55 55 57 57 57 ...
                    57 57 59 59 61 63 64 66 64 66 66 68 ...
                    66 66 66 68 68 68 66 66 66 66 64 66 ...
                    64 64 64 63 63 61 61 61 61 59 59 59 ...
                    59  59  59  59];

    case 6 % Friday 08-Nov-2019 00:00:00, 24hrs
        date_time='8-Nov-2019 00:00:00'; %datenum=737737
        % https://www.wunderground.com/history/daily/gb/newcastle-upon-tyne/EGNT/date/2019-11-8
        daily_temp=[43 43 43 41 41 41 41 41 41 39 39 41 ...
                    41 39 39 39 39 39 39 39 39 39 41 41 ...
                    41 41 41 41 39 39 39 39 39 37 39 39 ...
                    37 37 37 36 37 37 36 37 37 36 34 34 ...
                    32  34  32  32];
end

dt=datenum(datetime(date_time));
```

```
77  assignin('base','dt',dt);
78
79  if (ge(date_time_option,3))
80       set_param([ 'optim_ctrl_model_sim/building_subsystem/Daily Temp Variation/'...
81       'select_temp_out'],'Value','0')
82  else
83       set_param([ 'optim_ctrl_model_sim/building_subsystem/Daily Temp Variation/'...
84       'select_temp_out'],'Value','1')
85  end
86
87  %% Set Power System Parameters
88
89  R=0.05;
90  Tg=0.25;
91  Tt=0.6;
92  H=5;
93  D=0.8;
94  Ki=0.2;
95  MVA=300;
96  Th=9.65;
97  Kh=1.16;
98
99  %% Set SOC Model Parameters
100
101 SDR=0.017;
102 tau=2000;
103 tou_weekday=[4.99 11.99 24.99 11.99 4.99];
104
105 %% Set Outdoor Temperature Variation
106
107 tmd_dt=datetime(date_time);
108 tmd_sofd=dateshift(tmd_dt,'start','day');
109 tmd_Δ=between(tmd_sofd,tmd_dt);
110 tmd_sec=seconds(time(tmd_Δ));
111 tmd_phase=0.000072722*tmd_sec;
112
113 %% Set Building Parameters
114
115 r2d=180/pi; % converst radians to degrees
116 t1=1;
117
118 % Building
119 lenHouse=30; % House length = 30 m
120 widHouse=10; % House width = 10 m
121 htHouse=4; % House height = 4 m
122 pitRoof=40/r2d; % Roof pitch = 40 deg
123 numWindows=6; % Number of windows = 6
124 htWindows=1; % Height of windows = 1 m
125 widWindows=1; % Width of windows = 1 m
126 windowArea=numWindows*htWindows*widWindows;
127 wallArea=2*lenHouse*htHouse + 2*widHouse*htHouse + ...
128             2*(1/cos(pitRoof/2))*widHouse*lenHouse + ...
129             tan(pitRoof)*widHouse - windowArea;
130
131 % Insulation
132 % Glass wool in the walls, 0.2 m thick
133 % k is in units of J/sec/m/C - convert to J/hr/m/C multiplying by 3600
134 kWall=0.038*t1;   % hour is the time unit
135 LWall=0.2;
136 RWall=LWall/(kWall*wallArea);
137 % Glass windows, 0.01 m thick
138 kWindow=0.78*t1;  % hour is the time unit
139 LWindow=0.01;
140 RWindow=LWindow/(kWindow*windowArea);
141
142 % Equivalent thermal resistance for the whole building
143 Req=RWall*RWindow/(RWall + RWindow);
144 % c = cp of air (273 K) = 1005.4 J/kg-K
145 c=1005.4;
146
147 % Temperature of the heated air (degC)
148 THeater=50;
149 % Air flow rate Mdot = 1 kg/sec = 3600 kg/hr
150 Mdot=t1;  % hour is the time unit
151
152 % Total internal air mass = M
153 % Density of air at sea level = 1.2250 kg/m^3
154 densAir = 1.2250;
155 M=(lenHouse*widHouse*htHouse+tan(pitRoof)*widHouse*lenHouse)*densAir;
156
157 % Cost of energy storage system expressed in tems of energy capacity cost
```

```
158 % (GBP spent per unit of total energy stored as expressed in GBP per
159 % kilowatthours. Assume all electric energy is transformed to heat energy.
160 ess_cost_kWh=0.0199;
161
162 % Set initial indoor temperature = 18 deg C
163 TinIC=18;
```

Listing D.9 optim_ctrl_model_data.m

D.15 prepare_aux_data.m

```
function [ctrl_action]=prepare_aux_data(path)
%% MATLAB Function Description
%
%  Title: Prepare Auxcilary Data
%  Filename: prepare_aux_data.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function converts shortest path represented as nodemap (11x25)
%  index to temperature (control action) for each stage.
%  Called from fcn visual_group_path to create 4-hour scrolling figure.
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Convert Nodemap to Control Action

% define temperature range
t0tempSP=[20.5 20 19.5 19 18.5 18 17.5 17 16.5 16 15.5];

% initialise variable
ctrl_action=zeros(1,25);

% Set control action for each stage
for j=1:size(path,1)
    for ctrl_stage=1:25
        ctrl_action(j,ctrl_stage)=t0tempSP(path(j,ctrl_stage)-(11*(ctrl_stage-1)));
    end
end
```

Listing D.10 prepare_aux_data.m

D.16 prepare_comfort_values.m

```
function [tempSP]=prepare_comfort_values(temp)
%% MATLAB Function Description
%
%  Title: Prepare Comfort Values
%  Filename: prepare_comfort_values.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function sets temperature setpoint (control action) depending on
%  measured temperature. System limited to operate in temperature range
%  15.5degC (minimum) to 20.5degC (maximum).
%

%% Change History
%
%   1. [06-11-2019] Initial
%   2. [21-01-2020] Set upper temperature range to 20.5 irrespective of
%   measured temperature (last line in ifelse block set to 23.00)

%% Set Temperature Setpoint

if (temp ≥ 12) && (temp < 15.75)
    tempSP=15.5;
elseif (temp ≥ 15.75) && (temp < 16.25)
    tempSP=16;
elseif (temp ≥ 16.25) && (temp < 16.75)
    tempSP=16.5;
elseif (temp ≥ 16.75) && (temp < 17.25)
    tempSP=17;
elseif (temp ≥ 17.25) && (temp < 17.75)
    tempSP=17.5;
elseif (temp ≥ 17.75) && (temp < 18.25)
    tempSP=18;
elseif (temp ≥ 18.25) && (temp < 18.75)
    tempSP=18.5;
elseif (temp ≥ 18.75) && (temp < 19.25)
    tempSP=19;
elseif (temp ≥ 19.25) && (temp < 19.75)
    tempSP=19.5;
elseif (temp ≥ 19.75) && (temp < 20.25)
    tempSP=20;
elseif (temp ≥ 20.25) && (temp < 28.00)
    tempSP=20.5;
end

%end prepare_comfort_values
```

Listing D.11 prepare_comfort_values.m

D.17 prepare_digraph.m

```
function [G,B]=prepare_digraph(gridmap,edgepage)
%% MATLAB Function Description
%
%  Title: Prepare Digraph
%  Filename: prepare_digraph.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function prepares digraph, transposing gridmap (31x72) to
%  edgelist (733x3) before creating sparse adjacency matrix and finally full
%  adjacency matrix. The start and end nodes and their respective edge
%  weights format is prepared for fcn dijkstra.
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Prepare Digraph

% Convert 31x72 gridmap to a 733x3 edgelist
Y=gridmap(:,1:3,edgepage);
for i=2:24
    Y=cat(1,Y,gridmap(:,((i*3)-2:i*3),edgepage));
end

% Create array for node names {1,2,...,275}
name={};
for i=1:275
    name=cat(2,name,num2str(i));
end

% Define s=source, t=target, and w=edge weight
s=Y(:,1)';
t=Y(:,2)';
w=Y(:,3)';

% Define digraph G
G=digraph(s,t,w,name);

% Convert digraph into sparse adjacency matrix
A=adjacency(G,'Weighted');

% Convert sparse matrix to full matrix
B=full(A);

%end prepare digraph
```

Listing D.12 prepare_digraph.m

D.18 prepare_dv_values.m

```
function [dv,dv_gridmap,dv_nodepath,dv_rescale]=prepare_dv_values(horizon,S0_date)
%% MATLAB Function Description
%
%  Title: Prepare Demand Values
%  Filename: prepare_dv_values.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function computes demand values for duration of horizon window at
%  sample rate of 10 minutes; one per stage. Formating for scrolling
%  figure including rescaled, gridmap and nodepath included.
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Compute Demand Values

% Set rescale parameters
lower=1;
upper=11;
inmin=0;
inmax=1;
dv=zeros(horizon,1); % initialise array of all zeros
dv_rescale=zeros(horizon,1); % initialise array of all zeros

% Compute demand value (dv) set for each stage. Set includes
%       dv: Spline Coefs
%       dv_rescaled: dv(rescaled)
%       dv_gridmap: dv(gridmap)
%       dv_nodepath: dv(node path)
for n=0:horizon-1
    Sn_date=datetime(S0_date,'ConvertFrom','datenum')+minutes(10*n);
    dv(n+1,:)=demand(Sn_date); % Spline Coefs
    dv_rescale_single=lower+(dv(n+1,1)-inmin)./(inmax-inmin).*(upper-lower);
    dv_rescale(n+1,:)=ceil(dv_rescale_single); % dv(rescaled)
    dv_gridmap=12.-dv_rescale; % dv(gridmap)
    dv_nodepath=dv_rescale; % dv(node path)
end

%end prepare_dv_values
```

Listing D.13 prepare_dv_values.m

D.19 prepare_edgepath.m

```
function [edgepath]=prepare_edgepath(gridmap,edgepage,path)
%% MATLAB Function Description
%
%  Title: Prepare Edgepath
%  Filename: prepare_edgepath.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function returns list of numbers that describes edgepath
%  between start and end nodes of each stage.
%  For each start and end node pair code searches for index from nodemap
%  (11x25) at each stage. The intercept is the edgepath; starting from
%  S1 to S25.

%% Change History
%
%   1. [06-11-2019] Initial

%% Compute Edgepath

% For each stage, compute edgepath
edgepath=[];
for i=1:24
    path_s=find(gridmap(:,(i*3)-2,edgepage)==path(i));
    path_t=find(gridmap(:,(i*3)-1,edgepage)==path(i+1));
    int=intersect(path_s,path_t);
    edgepath=cat(2,edgepath,sub2ind(size(gridmap(:,:,edgepage)),int,i));
end

%end prepare_edgepath
```

Listing D.14 prepare_edgepath.m

D.20 prepare_gridmap.m

```
function [t0minidx,t240minidx,gridmap] = prepare_gridmap(nodepath,gridmap,edgepage,edgepath)
%% MATLAB Function Description
%
%  Title: Prepare Gridmap
%  Filename: prepare_gridmap.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function starts with gridmap (31x72x4) template. Maps nodepath
%  (11x25) onto gridmap (31x72) for each objective function (page):
%  comfort, demand and tou (tariff).
%  At each stage the min value is defined as the stage centroid,
%  all remaining values are populated, increasing/decreasing in
%  value moving up/down in the same col (stage). The index where the
%  min value at t0 and t240 is found are stored in t0minidx and t240minidx
%  respectively.
%  Exception handling at boundary upper and lower is included.
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Prepare Gridmap

i=1; % set count variable (ensure correct increase/decrease in temperature)
n=1;
col=3; % set column number
stage_centroid=1; % set stage centroid value
node=nodepath(n,:);

% calculate start node index
nodeidxs=(node*2)+(node-2);

% set the target node index the same as the start node index
nodeidxt=nodeidxs;

% dv at S1 to S24
% =================
% set centroid value for each stage and increase remaining values at stage
% moving away from stage centroid until reach min and max boundary
% (vertically)
for n=1:24
    node=nodepath(n,:);
    nodeidxs=(node*2)+(node-2);

    % if start node index is greater than target node index...
    if nodeidxs>nodeidxt
        if (i<2)
            gridmap(nodeidxt+1,(n-1)*col,edgepage)=stage_centroid;
            i=i+1;
            for j=nodeidxt:-1:1
                gridmap(j,(n-1)*col,edgepage)=stage_centroid+nodeidxt+1-j;
            end

            for j=nodeidxt+2:1:31
                gridmap(j,(n-1)*col,2)=stage_centroid+j-nodeidxt-1;
            end

            % determine value and idx of min value from previous stage
            [~,idx]=min(gridmap(:,col*(n-1),edgepage));
             % maintain list of edgepath of shortest path in grid map
            edgepath(:,end)=sub2ind(size(gridmap(:,:,edgepage)),idx,(n-1));
        end
    else
        i=1;
    end

    % if start node index is less than target node index...
    if nodeidxs<nodeidxt
        if (i<2)
            gridmap(nodeidxt-1,(n-1)*col,edgepage)=stage_centroid;
            i=i+1;
             for j=nodeidxt-2:-1:1
                gridmap(j,(n-1)*col,edgepage)=stage_centroid+nodeidxt-1-j;
             end
```

```
77
78                  for j=nodeidxt:1:31
79                      gridmap(j,(n-1)*col,edgepage)=stage_centroid+j-nodeidxt+1;
80                  end
81
82                 % determine value and idx of min value from previous stage
83                  [¬,idx]=min(gridmap(:,col*(n-1),edgepage));
84                   % maintain list of edgepath of shortest path in grid map
85                  edgepath(:,end)=sub2ind(size(gridmap(:,:,edgepage)),idx,(n-1));
86              end
87          else
88              i=1;
89          end
90
91          % if start node index equals the target node index...
92          gridmap(nodeidxs,n*col,edgepage)=stage_centroid;
93          for j=nodeidxs-1:-1:1
94              gridmap(j,n*col,edgepage)=stage_centroid+nodeidxs-j;
95          end
96          for j=nodeidxs+1:1:31
97              gridmap(j,n*col,edgepage)=stage_centroid+j-nodeidxs;
98          end
99          nodeidxt=nodeidxs;
100
101         % determine value (not required) and idx of min value from previous stage
102         [¬,idx]=min(gridmap(:,col*n,edgepage));
103         % maintain list of edgepath of shortest path in grid map
104         edgepath=cat(2,edgepath,sub2ind(size(gridmap(:,:,edgepage)),idx,n));
105     end
106
107 % compute value (not required) and index (row number {1,2,...,11}) showing
108 % lowest value at S1 and S24 values are use when plotting shortest path
109  [¬,t0minidx]=min(gridmap(:,3,edgepage));
110  [¬,t240minidx]=min(gridmap(:,72,edgepage));
111
112 %end prepare gridmap
```

Listing D.15 prepare_gridmap.m

D.21 prepare_tc_gridmap.m

```
function [tcv,tcvdata,gridmap,t0minidx,t240minidx]=prepare_tc_gridmap ...
    (t0tempSP,tempSP,S0_date,gridmap,edgepath,edgepage,mintempthresholdparam)
%% MATLAB Function Description
%
%  Title: Prepare Thermal Comfort Gridmap
%  Filename: prepare_tc_gridmap.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function starts with gridmap (31x72x4) template. Maps nodepath
%  (11x25) onto gridmap (31x72) for each objective function (page):
%  comort, demand and tou (tariff).
%  At each stage the min value is defined as the stage centroid,
%  all remaining values are populated, increasing/decreasing in
%  value moving up/down in the same col (stage). The index where the
%  min value at t0 and t240 is found are stored in t0minidx and t240minidx
%  respectively.
%  Exception handling at boundary upper and lower is included.
%
%  Similar to prepare_gridmap.m but specific to thermal comfort.
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Prepare Thermal Comfort Gridmap

% Define stage S1 column number used in grid template, multiples of col is
% used to calculate S2 to S24
col=3;

% Find index number of value in array t0tempSP that matches the recorded
% temperature measurement
node=find(t0tempSP==tempSP);

% Calculate node index of declared temperature setpoint at t0; (2016≤nodeidx=<1)
nodeidx=(node*2)+(node-2);

% tc at S1
% ========
% Calculate thermal comfort value (tcv) edge
% tcv=comfort(stime,1) where stime=[10,20,...,n] minutes
% no difference if users thermal comfort calc_mean is COOL or COLD. Also no
% difference to control action if users thermal comfort calc_mean is WARM
% or HOT. Returns [n1 n2 n3] where n1=occupants, n2=response, n3=calc_mode
Sn_date=datetime(S0_date,'ConvertFrom','datenum');
tcv=comfort_2(Sn_date,1);
tcvdata(1,:)=tcv;

% Set initial edge weight (S1)
% Check number of op. If op=0 then set edge weight to direct path
% to reduce tempSP by 0.5degC. Otherwise set edge weight based on
% return fcn: comfort value.
% If nil op then set path to reduce tempSP by 0.5degC from S0 to S1
if (tcv(1)==0)
    a=1;b=0.5;c=0.25;
else
    % tc=-1 or -2 indicating too hot, reduce tempSP. Least value path
    % is to lower temp
    if (tcv(3)<0)
            a=1;b=0.5;c=0.25;
    % tc=0 indicating okay, maintain tempSP. Least value path is to
    % same temp
    elseif (tcv(3)==0)
            a=0.5;b=0.25;c=0.5;
    % tc-1 or 2 indicating too cold, increase tempSP, Least value path to
    % higher temp.
    elseif (tcv(3)>0)
            a=0.25;b=0.5;c=1.0;
    end
end

% Check for outliner temperature values (20.5 and 15.5) and restrict setting
% of edge weights to valid edges, ie not possible to assign edge weight
% from 20.5 to 21.0.
```

```
77  % higher sn<tn (source node is less than targe node)
78  if (nodeidx-1==0)
79      gridmap(nodeidx,col,edgepage)=b;
80      gridmap(nodeidx+1,col,edgepage)=c;
81  % lower sn>tn
82  elseif (nodeidx-31==0)
83      gridmap(nodeidx-1,col,edgepage)=a;
84      gridmap(nodeidx,col,edgepage)=b;
85  % equal sn=tn
86  else
87      gridmap(nodeidx-1,col,edgepage)=a;
88      gridmap(nodeidx,col,edgepage)=b;
89      gridmap(nodeidx+1,col,edgepage)=c;
90  end
91
92  % Find row number (t0minidx) listing minimum edge weight at S1 use t0minidx
93  % in dijkstra.mlx to set source node when calculating shortest path
94  [t0value,t0minidx]=min(gridmap(:,3,edgepage));
95
96  % tc at S2 to S24
97  % ===============
98  for n=1:23 %23
99      % Display label > 'Stage: n (k)' where n=[2,3,...,24], k=[6,9,...,72]
100     % column number disp(['Stage: ',num2str(n+1),' (',num2str((n*3)+3),') '])
101
102     Sn_date=datetime(S0_date,'ConvertFrom','datenum')+minutes(10*n);
103
104     % Calculate tc for next stage
105     tcv=comfort_2(Sn_date,0);
106     tcvdata(n+1,:)=tcv;
107
108     % Determine value (not required) and idx of min value from previous stage
109     [¬,idx]=min(gridmap(:,col*n,edgepage));
110
111     % Determine the previous stage target node
112     tnode=gridmap(idx,(col*n)-1,edgepage);
113
114     % Now declare the start node of next stage the target node from
115     % previous stage
116     snode=tnode;
117
118     % Determine the node index number of min value in previous stage
119     nodeidx=sub2ind(size(gridmap(:,:,edgepage)),idx,col*n);
120
121     % Find the index numbers of the start nodes in the next stage
122     % (with exception to outliners, there are three values returned)
123     [value,¬]=find(gridmap(:,1+(col*n),edgepage)==snode);
124
125     % Maintain list of edgepath of shortest path in grid map
126     edgepath=cat(2,edgepath,sub2ind(size(gridmap(:,:,edgepage)),idx,n));
127
128     % Check if op>0
129     if(tcv(1)>0)
130         % This ensures 1:12 maintains 12:23, 23:34 etc
131         if (value(1)==1)
132             gridmap(value(1),3+(col*n),edgepage)=t0value;
133         else
134             gridmap(value(2),3+(col*n),edgepage)=t0value;
135         end
136     else
137         % This ensures 1:n starts to fall when no occupancy
138         if (value(1)==1)
139             gridmap(value(1)+1,3+(col*n),edgepage)=t0value;
140         else
141             % Check path exceeds declared minimum temperature threshold value
142             if (idx<mintempthresholdparam)
143                 % Continue to force shortest path to lower temperature
144                 % if value is less than declared minimum temperature
145                 % threshold value
146                 gridmap(value(2)+1,3+(col*n),edgepage)=t0value;
147             else
148                 % Maintain shortest path on minimum temperate threshold
149                 % value if calculated temperature is less than minimum
150                 % temperature threshold value
151                 gridmap(value(2),3+(col*n),edgepage)=t0value;
152             end
153         end
154     end
155 end
156
157 % Compute row number showing lowest value at S24
```

```
158 % values are used when plotting shortest path
159  [¬,t240minidx]=min(gridmap(:,72,edgepage));
160
161 % Add final node to edgepath list that informs shortest path
162  edgepath(1,end+1)=(23*31)+t240minidx;
163
164 %end prepare tc gridmap
```

Listing D.16 prepare_tc_gridmap.m

D.22 prepare_tou_values.m

```
function [touv_gridmap,touv_nodepath,touv_rescale,touv]=prepare_tou_values ...
    (horizon,S0_date,period,tou_tariff)
%% MATLAB Function Description
%
%  Title: Prepare Time Of Use Values
%  Filename: prepare_tou_values.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function computes tou value set for S1 to S24 at every
%  10 minute interval specific to tou tariff and time of day.

%% Change History
%
%   1. [06-11-2019] Initial

%% Prepare TOU Values
% Set scale feature parameters
lower=3;
upper=9;
inmin=0.0499;
inmax=.2499;
% Initialise arrays of all zeros
touv=zeros(horizon,1);
touv_rescale=zeros(horizon,1);
% Compute tou value (touv) set for each stage. Set includes
% (1) touv: tou value, (2) touv_rescale: tou(rescaled),
% (3) touv_gridmap: tou(gridmap), (4) touv_nodepath: tou(node path).
for n=0:horizon
    Sn_date=datetime(S0_date,'ConvertFrom','datenum')+minutes(10*n);
    tod=timeofday(Sn_date);
    % For each weekend day
    if (isweekend(Sn_date))
        if ge(tod,period{1}) && lt(tod,period{2})
            touv(n+1,:)=tou_tariff(1);
        elseif ge(tod,period{2}) && lt(tod,period{5})
            touv(n+1,:)=tou_tariff(2);
        elseif ge(tod,period{5}) && lt(tod,period{6})
            touv(n+1,:)=tou_tariff(1);
        end
    % For each week day
    else
        if ge(tod,period{1}) && lt(tod,period{2})
            touv(n+1,:)=tou_tariff(1);
        elseif ge(tod,period{2}) && lt(tod,period{3})
            touv(n+1,:)=tou_tariff(2);
        elseif ge(tod,period{3}) && lt(tod,period{4})
            touv(n+1,:)=tou_tariff(3);
        elseif ge(tod,period{4}) && lt(tod,period{5})
            touv(n+1,:)=tou_tariff(2);
        elseif ge(tod,period{5}) && lt(tod,period{6})
            touv(n+1,:)=tou_tariff(1);
        end
    end
end
% Apply feature scaling
for n=0:size(touv(:),1)-1
    touv_rescale_single=lower+(touv(n+1,1)-inmin)./(inmax-inmin).*(upper-lower);
    touv_rescale(n+1,:)=ceil(touv_rescale_single); % touv(rescaled)
end
% Define touv gridmap
touv_gridmap=12.-touv_rescale; % touv(gridmap)
% Define touv nodepath
touv_nodepath=touv_rescale; % touv(node path)

%end prepare tou values
```

Listing D.17 prepare_tou_values.m

D.23 soc.m

```
function soc(block)
%% SOC
%
%  Title: State of Charge
%  Filename: soc.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  Code tagged to Simulink model block soc_model.
%  Operates in 2 Modes: [0] normal operations [1] demand event
%  Initially SOC assumed 0 and will start to charge. At high
%  threshold ESS declared available for use (FIT=1). FIT status revert
%  back to 0 when low threshold reached (on discharge).
%  If SOC available and tariff HIGH (level 3), power switch to ESS (PWR=1).
%  When tariff LOW (level 1 or 2) power switch to GRID (PWR=0).
%  On receipt of demand event signal, MODE=0. Priority sets
%  ESS to charge during 4-hour ramp time before demand event starts and
%  power switch to GRID (PWR=0). At demand event start power switch to ESS
%  (PWR=1), ESS begins to discharge. Maintain ESS power for duration of
%  demand event. At end of demand event revert back to normal operations
%  (MODE=0). Self-Discharge Rate (SDR) applies on discharge.
%

setup(block);

%endfunction: soc(block)

function setup(block)
%% Setup Functional Port Properties

% Register number of ports
block.NumInputPorts  = 4;
block.NumOutputPorts = 2;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Override input port properties
block.InputPort(1).Dimensions        = 1;  % DIR
block.InputPort(1).DatatypeID        = 8;  % boolean
block.InputPort(1).Complexity        = 'Real';
block.InputPort(1).DirectFeedthrough = true;

% Override input port properties
block.InputPort(2).Dimensions        = 1;  % DATA
block.InputPort(2).DatatypeID        = 0;  % double
block.InputPort(2).Complexity        = 'Real';
block.InputPort(2).DirectFeedthrough = true;

% Override input port properties
block.InputPort(3).Dimensions        = 1;  % t_mode
block.InputPort(3).DatatypeID        = 0;  % double
block.InputPort(3).Complexity        = 'Real';
block.InputPort(3).DirectFeedthrough = true;

% Override input port properties
block.InputPort(4).Dimensions        = 1;  % MODE
block.InputPort(4).DatatypeID        = 0;  % double
block.InputPort(4).Complexity        = 'Real';
block.InputPort(4).DirectFeedthrough = true;

% Override output port properties
block.OutputPort(1).Dimensions       = 1;  % FIT
block.OutputPort(1).DatatypeID       = 0;  % double
block.OutputPort(1).Complexity       = 'Real';

% Override output port properties
block.OutputPort(2).Dimensions       = 1;  % PWR
block.OutputPort(2).DatatypeID       = 0;  % double
block.OutputPort(2).Complexity       = 'Real';

% Register parameters
block.NumDialogPrms = 0;

% Register sample times
```

```
77  block.SampleTimes = [30 0];
78
79  % Specify the block simStateCompliance to default
80  block.SimStateCompliance = 'DefaultSimState';
81
82  % Register nethods
83  block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
84  block.RegBlockMethod('Start', @Start);
85  block.RegBlockMethod('Outputs', @Outputs); % required
86  block.RegBlockMethod('Update', @Update);
87  block.RegBlockMethod('Terminate', @Terminate); % required
88  block.RegBlockMethod('SetInputPortSamplingMode',@SetInpPortFrameData);
89
90  %endfunction: setup(block)
91
92  function DoPostPropSetup(block)
93
94  % Initialise the Dwork vectors
95  block.NumDworks = 1;
96
97  % DWork(1) store value at input port 2 [DATA] = raw SOC
98  block.Dwork(1).Name             = 'D1';
99  block.Dwork(1).Dimensions       = 1;
100 block.Dwork(1).DatatypeID       = 0;      % double
101 block.Dwork(1).Complexity       = 'Real'; % real
102 block.Dwork(1).UsedAsDiscState = true;
103
104 %endfunction: DoPostPropSetup(block)
105
106 function Start(block)
107 %% Set Start Conditions
108
109 % Assign Dwork(1) to 0
110 block.Dwork(1).Data = 0;
111
112 %endfunction: Start(block)
113
114 function Outputs(block)
115 %% Outputs
116
117 % define model paths
118 path_1='optim_ctrl_model_sim/scheduler_subsystem/ess_subsystem/SOC_hold';
119 path_2='optim_ctrl_model_sim/building_subsystem/ESS Cost Enable';
120 path_3='optim_ctrl_model_sim/scheduler_subsystem/ess_subsystem/CD';
121
122 % Determine MODE: [0]=normal, [1]=demand event (ramp plus duration)
123 if (block.InputPort(4).Data==0)
124     % Normal Operations
125     set_param(path_1,'Value','1')
126     if (block.InputPort(1).Data==1) % DIR increasing (charge)
127         if (block.InputPort(2).Data>0.8) % detect SOC > 0.8
128             block.OutputPort(1).Data = 1; % FIT=1
129             if (block.InputPort(3).Data==3) % detect high tariff
130                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
131                 set_param(path_2,'Value','1')
132                 set_param(path_3,'Value','0')
133                 set_param(path_1,'Value','1')
134             else
135                 block.OutputPort(2).Data = 0; % PWR=0 (GRID)
136                 set_param(path_2,'Value','0')
137                 set_param(path_3,'Value','1')
138                 set_param(path_1,'Value','1')
139             end
140         else
141             block.OutputPort(2).Data = 0; % PWR=0
142             set_param(path_2,'Value','0')
143             set_param(path_3,'Value','1')
144             set_param(path_1,'Value','1')
145         end
146     else % DIR decreasing (discharge)
147         if (block.InputPort(2).Data<0.2) % detect SOC < 0.2
148             block.OutputPort(1).Data = 0; % FIT=0
149             block.OutputPort(2).Data = 0; % PWR =0
150             set_param(path_2,'Value','0')
151             set_param(path_3,'Value','1')
152             set_param(path_1,'Value','1')
153             return
154         else
155             if (block.InputPort(3).Data==3) % detect high tariff
156                 block.OutputPort(2).Data = 1; % PWR=1
157                 set_param(path_2,'Value','1')
158                 set_param(path_1,'Value','1')
```

```
159             else
160                 block.OutputPort(2).Data = 0; % PWR=0
161                 set_param(path_2,'Value','0')
162                 set_param(path_1,'Value','0');
163             end
164         end
165     end
166 else
167     % Demand Event (active for ramp plus duration)
168     if (block.InputPort(1).Data==1) % DIR increasing (charge)
169         if (block.InputPort(2).Data>0.90) % detect SOC > 0.95
170             block.OutputPort(1).Data = 1; % FIT=1
171             if (block.InputPort(3).Data==3) % detect high tariff
172                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
173                 set_param(path_2,'Value','1')
174                 set_param(path_3,'Value','0')
175                 set_param(path_1,'Value','1')
176             else
177                 block.OutputPort(2).Data = 0; % PWR=0 (GRID)
178                 set_param(path_2,'Value','0')
179                 set_param(path_3,'Value','1')
180                 set_param(path_1,'Value','1')
181             end
182         else
183             if (block.InputPort(3).Data==3) % detect high tariff
184                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
185                 set_param(path_2,'Value','1')
186                 set_param(path_3,'Value','0')
187                 set_param(path_1,'Value','1')
188             else
189                 block.OutputPort(2).Data = 0; % PWR=0
190                 set_param(path_2,'Value','0')
191                 set_param(path_3,'Value','1')
192                 set_param(path_1,'Value','1')
193             end
194         end
195     else % DIR decreasing (discharge)
196         if (block.InputPort(2).Data<0.2) % detect SOC < 0.2
197             block.OutputPort(1).Data = 0; % FIT=0
198             block.OutputPort(2).Data = 0; % PWR =0
199             set_param(path_2,'Value','0')
200             set_param(path_3,'Value','1')
201             set_param(path_1,'Value','1')
202             return
203         else
204             if (block.InputPort(3).Data==3) % detect high tariff
205                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
206                 set_param(path_2,'Value','1')
207                 set_param(path_1,'Value','1')
208             else
209                 block.OutputPort(2).Data = 0; % PWR=0 (GRID)
210                 set_param(path_2,'Value','0')
211                 set_param(path_1,'Value','1');
212                 set_param(path_3,'Value','1')
213             end
214         end
215     end
216 end
217
218 %endfunction: Outputs(block)
219
220 function Update(block)
221 %% Update Dwork
222
223 % Update Dwork(1) to InputPort(2) [Data] = raw SOC
224 block.Dwork(1).Data = block.InputPort(2).Data;
225
226 %endfunction: Update(block)
227
228 function SetInpPortFrameData(block, idx, fd)
229
230 % Set the sampling of the input ports
231 block.InputPort(idx).SamplingMode=fd;
232 for i=1:block.NumOutputPorts
233     block.OutputPort(i).SamplingMode=fd;
234 end
235
236 %endfunction: SetInpPortFrameData(block)
237
238 function Terminate(block)
239
```

```
240 %endfunction: Terminate(block)
```

Listing D.18 soc.m

D.24 tariff_mode.m

```
function [t_mode] = tariff_mode (tariff)
%%TARIFF_MODE computes tariff mode value
%
%  Construction:
%   [T_MODE] = TARIFF_MODE(TARIFF) compares tariff at given time of day
%   against set criteria. Model set criteria includes two levels:
%         Level 1: 4.99
%         Level 2: 11.99
%   If tariff is greater than zero and less than or equal to Level 1 then
%   set the tariff mode to 1 (t_mode=1).
%   If tariff is greater than Level 1 and less than or equal to Level 2
%   then set the tariff mode to 2 (t_mode=2)
%   If tariff is greater than Level 2 (assumed to be highest tariff band)
%   then set the tariff mode to 3 (t_mode=3)

%% MATLAB Function Description
%
%  Title: Tariff Mode
%  Filename: tariff_mode.m
%  Prepared by: Sean Williams
%  Date: 1 Aug 2019
%
%  MATLAB function compute tariff mode value
%

%% Change History
%
%   1. [01-08-2019] Initial

%% Assign Tariff Mode

 if (tariff(1,1)>0) && (tariff(1,1)≤tariff(2,1))
     t_mode=1;
 elseif (tariff(1,1)>tariff(2,1)) && (tariff(1,1)≤tariff(3,1))
     t_mode=2;
 elseif (tariff(1,1) > tariff(2,1))
     t_mode=3;
 else
     t_mode=0;
 end
```

Listing D.19 tariff_mode.m

D.25 visual_comfort_data.m

```
function visual_comfort_data(horizon,tcvdata,t1)
%% MATLAB Function Description
%
%  Title: Visual Comfort Data
%  Filename: visual_comfort_data.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function creates scrolling 4-hour window showing comfort data.
%  Figure is fixed position: subplot(1,4,1). Data has been formated from
%  showing series of static comfort shortest path on gridmap (31x72) plots
%  to scrolling gridmap. Data includes:
%
% # Comfort Data: Number of Occupant
% # Comfort Data: Number of Responses
% # Comfort Data: Comfort
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Visual Comfort Data

%set(0,'DefaultFigureVisible','on');
% Create new figure and set view options
figure(1)
set(figure(1),'Name','Comfort Data','NumberTitle','off','ToolBar','none');

% Set position to top left quadrant
movegui(figure(1),[400,580])

% Set time data to x-axis
S0_date=datetime(datestr(evalin('base','dt')));
t5=t1+minutes(230);
t45=t1:minutes(10):t5;

% p1=plot comfort data: number of occupants - color blue
p1=stairs(t45,tcvdata(:,1),'color',[0 102/255 204/255],'LineWidth',1.2,'LineStyle','-');
hold on

% p2=plot comfort data: number of responses - color green
p2=stairs(t45,tcvdata(:,2),'color',[0 153/255 0],'LineWidth',1.2,'LineStyle','-');

% p3=plot comfort data: comfort - color red
p3=stairs(t45,tcvdata(:,3),'color',[153/255 0 0],'LineWidth',1.5,'LineStyle','-');
hold off
refreshdata

% Format
lgd={'Occupants','Response','Comfort'};
lgd=legend([p1 p2 p3],lgd);
title(lgd,['Comfort Info (', char(t1,'HH:mm'),')']);
title(['Comfort (' num2str(ceil((horizon-1)/6)) ' hrs): ', char(t1,'dd-MMM-yyyy HH:mm')]);
xticks(t45(1:3:end));
xtickformat('HH:mm')
ylabel('gridmap');
axis tight
ylim([-5 105]);
grid on
box on

%end visual comfort data
```

Listing D.20 visual_comfort_data.m

D.26 visual_demand_data.m

```
function visual_demand_data(horizon,dv,dv_gridmap,dv_rescale,t1)
%% MATLAB Function Description
%
%  Title: Visual Demand Data
%  Filename: visual_demand_data.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function creates scrolling 4-hour window showing demand data.
%  Figure is fixed position: subplot(1,4,2). Data has been formated from
%  showing series of static demand shortest path on gridmap (31x72) plots
%  to scrolling gridmap. Data includes
%
% # Demand Value: Spine Coefs (l y-axis)
% # Demand Value: Rescale (r y-axis)
% # Demand Value: Gridmap (r y-axis)
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Visual Demand Data
% set(0,'DefaultFigureVisible','on');
% Create new figure and set view options
figure(2)
left_color = [0.1 0.1 0.1];
right_color = [0.1 0.1 0.1];
set(figure(2),'defaultAxesColorOrder',[left_color; right_color],...
      'Name','Demand Data','NumberTitle','off','ToolBar','none');
movegui(figure(2),[1000 580])
% Set time data to x-axis
S0_date=datetime(datestr(evalin('base','dt')));
t5=t1+minutes(240);
t45=t1:minutes(10):t5;
% p1=plot demand value Spline Coefs - colour green
p1=plot(t45,dv(:,1), 'LineWidth',1.5,'Color',[0.298,0.6,0]);
% Format left
ylabel('magnitude');
yyaxis right
% p2=plot demand value rescale - colour red
p2=stairs(t45,dv_rescale(:,1),'LineWidth',1.5,'Color',[0.753,0,0]);
hold on
% p3=plot demand value gridmap - color blue
p3=stairs(t45,dv_gridmap,'LineWidth',1.5,'color',[0/255 112/255 192/255],'LineStyle','-');
hold off
% Format right
ylim([1 11])
ylabel('gridmap');
% Format general
yyaxis left
xticks(t45(1:3:end));
xtickformat('HH:mm')
axis tight
ylim([0 1]);
grid on; box on; hold off
lgd={'Spline Coefs','dv (rescaled)','dv (gridmap)'};
lgd=legend([p1 p2 p3],lgd);
title(lgd,['Demand Info (', char(t1,'HH:mm'),')']);
title(['Demand (' num2str(ceil((horizon-1)/6)) ' hrs): ' char(t1,'dd-MMM-yyyy HH:mm')])

%end visual demand data
```

Listing D.21 visual_demand_data.m

D.27 visual_group_path.m

```
function visual_group_path(path_c, path_d, path_t, path_a, t1)
%% MATLAB Function Description
%
%  Title: Visual Group Path
%  Filename: visual_group_path.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function creates scrolling 4-hour window showing group path.
%  Figure is fixed position: subplot(1,4,4). Data has been formated from
%  showing series of static group shortest path on gridmap (31x72) plots
%  to scrolling gridmap. Group data has been calculated using k-means
%  method to find optimal shortest path based on comfort, demand and tou
%  tariff data. Data includes:
%
% # Comfort
% # Demand
% # TOU
% # Group
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Visual Group Path

%set(0,'DefaultFigureVisible','on');
% Create new figure and set view options
figure(4)
set(figure(4),'Name','Gridmap Data','NumberTitle','off','ToolBar','none');

% Set position to bottom right quadrant
movegui(figure(4),[1000,80])

% Convert data from gridmap index to temperature values
[ctrl_action_c]=prepare_aux_data(path_c(end,:));
[ctrl_action_d]=prepare_aux_data(path_d(end,:));
[ctrl_action_t]=prepare_aux_data(path_t(end,:));
[ctrl_action_a]=prepare_aux_data(path_a(end,:));

% Set time data to x-axis
S0_date=datetime(datestr(evalin('base','dt')));
t5=t1+minutes(240);
t45=t1:minutes(10):t5;

% p1=plot comfort data - color red
p1=plot(t45,ctrl_action_c(end,:),'Color',[153/255 0 0],...
    'LineWidth',1.5);
hold on

% p2=plot demand data - color blue
p2=plot(t45,ctrl_action_d(end,:),'Color',[0/255 112/255 192/255],...
    'LineWidth',1.5);

%p3=plot tou data - color magenta
p3=plot(t45,ctrl_action_t(end,:),'Color',[204/255 0/255 153/255],...
    'LineWidth',1.5);

%p4=plot group path - color green
p4=plot(t45,ctrl_action_a(end,:),'Color',[34/255 139/255 34/255],...
    'LineWidth',1.5);

% Format
xticks(t45(1:3:end));
xtickformat('HH:mm')
axis tight
ylim([14 21])
grid on
box on
hold off
refreshdata
lgd={'Comfort','Demand','TOU','Forecast'};
lgd=legend([p1 p2 p3 p4],lgd);
title(lgd,'Gridmap Data')
title(['S0\_date: ' char(t1, 'dd-MMM-yyyy HH:mm')]);
```

```
77
78 %end visual big path
```

Listing D.22 visual_group_path.m

D.28 visual_group_shortestpath.m

```
function visual_group_shortestpath(G_c,edgepath_c,G_d,edgepath_d,G_t,...
    edgepath_t,G_a,edgepath_a,t1,horizon)
%% MATLAB Function Description
%
%  Title: Visual Group Shortestpath
%  Filename: visual_group_shortestpath.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function creates static image of digraph, format changed to
%  show gridmap (31x72) view. Shortestpath for listed data shown.
%  If selected to view during Simulink model, new figure is created at end
%  of each cycle. To view 4-hour scrolling group data selec to view
%  'visual_group_path' as alternative. Data includes:
%
% # Comfort
% # Demand
% # TOU
% # Group
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Visual Static Group Shortestpath (Gridmap Format)

%gridmap_color=[0 0.533 0.8]; % colour - blue
gridmap_color=[0.745 0.745 0.745]; % colour - grey

% Create a figure and specify axis colours using default colour order
figure('Name','Optimisation Data','NumberTitle','off','ToolBar','none');

%op1=digraph of comfort gridmap - color red
op1=plot(G_c);
op1.NodeColor=[255/255 0/255 0/255];
op1.EdgeColor=gridmap_color;
hold on

%op2=digraph of demand gridmap - color blue
op2=plot(G_d);
op2.NodeColor=[0/255 112/255 192/255];
op2.EdgeColor=gridmap_color;

%op3=digraph of toue gridmap - color magenta
op3=plot(G_t);
op3.NodeColor=[204/255 0/255 153/255];
op3.EdgeColor=gridmap_color;

%op4=digraph of group gridmap - color green
op4=plot(G_a);
op4.NodeColor=[34/255 139/255 34/255];
op4.EdgeColor=gridmap_color;

% Plot pseudo gridmap for format only (forces nodes colour grey and
% enables legend to reflect highlighted paths op1 to op4.
op5=plot(G_a);
op5.NodeColor=gridmap_color;
op5.EdgeColor=gridmap_color;

% Highlight shortestpath for each data
highlight(op1,'Edges',edgepath_c,'Edgecolor',[153/255 0 0],...
    'LineWidth',2); % red
highlight(op2,'Edges',edgepath_d,'Edgecolor',[0/255 112/255 192/255],...
    'LineWidth',2); % blue
highlight(op3,'Edges',edgepath_t,'Edgecolor',[204/255 0/255 153/255],...
    'LineWidth',2); % magenta
highlight(op4,'Edges',edgepath_a,'Edgecolor',[34/255 139/255 34/255],...
    'LineWidth',2); % green

% Change layout of graph
layout(op1,'layered');
layout(op2,'layered');
layout(op3,'layered');
layout(op4,'layered');
layout(op5,'layered');
```

```
77
78 % Format
79  title  'TOT Gridmap  '
80
81 % use rotate to position plot correct left to right
82  op1.Parent.CameraUpVector=[-1,0,0];
83  op1.Parent.CameraPosition(3)=op1.Parent.CameraPosition(3)*-1;
84  op2.Parent.CameraUpVector=[-1,0,0];
85  op3.Parent.CameraUpVector=[-1,0,0];
86  op4.Parent.CameraUpVector=[-1,0,0];
87  op5.Parent.CameraUpVector=[-1,0,0];
88
89  title(['Optimisation Data (' num2str(ceil((horizon-1)/6))  ' hrs): ',...
90      char(t1,'dd-MMM-yyyy HH:mm')]);
91 legend([op1 op2 op3 op4],{'comfort','demand','tou','forecast'},...
92      'Location','south','NumColumns',4,'Box','off')
93
94 %end visual group shortestpath
```

Listing D.23 visual_group_shortestpath.m

D.29 visual_individual_shortestpath.m

```
function visual_individual_shortestpath(fig_name,G,edgepage,edgepath,t1,edgepath_color)
%% MATLAB Function Description
%
%  Title: Visual Individual Shortestpath
%  Filename: visual_individual_shortestpath.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function creates static image of individual digraph, format
%  changed to show gridmap (31x72) view. Shortestpath for individual data
%  shown. If selected to view during Simulink model, new figure is created
%  at end of each cycle.
%  To view 4-hour scrolling group data select to view respective
%  'visual_[data]_data' as alternative; [data]=[comfort|demand|tou].
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Visual Individual Shortestpath (Gridmap Format)

% Set time data to x-axis
t0=evalin('base','dt');
t0=datetime(datestr(t0))+minutes(10);

% Create figure and specify axis colours using default colour order
figure('Name',fig_name,'NumberTitle','off','ToolBar','none');

% Create diagram of gridmap, include formatting (grey|grey)
op1=plot(G,'NodeLabel',G.Nodes.Name,'EdgeLabel',G.Edges.Weight,...
     'EdgeFontWeight','normal',...
     'NodeLabelColor',[192/255,192/255,192/255],...
     'EdgeLabelColor',[160/255,160/255,160/255]);

% Highlight shortest path - colour specific to data
highlight(op1,'Edges',edgepath,'Edgecolor',edgepath_color,'LineWidth',3);

% Change layout of graph
layout(op1,'layered')

% Format
title([fig_name ' (p' num2str(edgepage) '): ' char(t1,'dd-MMM-yyyy HH:mm')])

% use rotate to position plot correct left to right
op1.Parent.CameraUpVector=[-1,0,0];
op1.Parent.CameraPosition(3)=op1.Parent.CameraPosition(3)*-1;

%end visual_individual_shortestpath
```

Listing D.24 visual_individual_shortestpath.m

D.30 visual_tou_data.m

```
function visual_tou_data(horizon,touv_rescale,t1)
%% MATLAB Function Description
%
%  Title: Visual Time of Use (Tariff) Data
%  Filename: visual_tou_data.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function creates scrolling 4-hour window showing tou data.
%  Figure is fixed position: subplot(1,4,3). Data has been formated from
%  showing series of static tou shortest path on gridmap (31x72) plots
%  to scrolling gridmap. Data includes
%
% # TOU Data: Rescaled
%

%% Change History
%
%   1. [06-11-2019] Initial

%% Visual TOU Data

%set(0,'DefaultFigureVisible','on');
% Create new figure and set view options
figure(3)
set(figure(3),'Name','TOU Data','NumberTitle','off','ToolBar','none');

% Set position to bottom left quadrant
movegui(figure(3),[400 80])

% Set time data to x-axis
S0_date=datetime(datestr(evalin('base','dt')));
t5=t1+minutes(250);
t45=t1:minutes(10):t5;

% p1=plot tou data - color magenta
p1=stairs(t45,touv_rescale(:,1), 'LineWidth',1.5,'Color',[204/255 0 153/255]);
refreshdata

% Format
lgd={'TOU'};
lgd=legend([p1],lgd);
title(lgd,['TOU Info (', char(t1,'HH:mm'),')']);
title(['TOU (' num2str(ceil((horizon-1)/6)) ' hrs): ', char(t1,'dd-MMM-yyyy HH:mm')]);
xticks(t45(1:3:end));
xtickformat('HH:mm')
ylabel('gridmap');
axis tight
ylim([1 11]);
grid on
box on

%end visual tou data
```

Listing D.25 visual_tou_data.m

D.31 Workspace variables (MAT-file)

The Simulink® model includes information about the surrounding environment that is useful to categorise into three groups. External conditions are monitored and trigger change in system behaviour when predefined conditions are satisfied. The information in the following tables represents the external conditions category and includes binary MATLAB® file that stores workspace variables (MAT-file).

Table D.5
Energy management model: grid4_1.mat[1]

Item	Name	Value	Parameter
1	grid4_0	$31 \times 72 \times 4$ double	-
2	grid4_0_1	31×72 double	-
3	grid_0	28×72 double	-

Table D.6
Energy management model: demand_info.mat

Item	Name	Value	Parameter
1	D48	`duration`	`D48=duration(0,0:30:1410,0)'`
2	mtwtfPAA_data	12×1 double	See Chapter 4 Table 4.1
3	mtwtfdata	1×48 double	[18.53 16.46 14.12 12.27 11.03 10.75 10.39 9.70 11.34 14.88 27.79 41.98 60.41 73.73 84.61 89.73 94.49 95.84 96.36 97.07 97.79 98.17 98.66 97.47 96.06 94.37 93.33 92.18 91.65 92.99 95.51 98.44 100.00 98.51 95.81 92.88 89.20 85.75 82.49 80.42 80.54 77.86 72.44 62.45 50.63 39.72 30.27 24.04]
4	ssPAA_data	12×1 double	See Chapter 4 Table 4.1
5	ssdata	1×48 double	[16.49 13.44 10.18 7.38 5.60 4.72 3.37 1.51 0.44 0.00 4.02 8.72 16.61 24.33 34.15 41.88 49.67 54.28 57.96 59.79 60.86 61.24 61.21 59.75 57.16 54.23 51.92 49.89 49.21 49.63 51.59 54.83 58.11 59.94 60.43 60.20 9.77 58.07 57.26 56.99 59.07 58.60 56.30 49.43 40.74 31.85 23.72 18.04]
6	t4	1×14 double	`{0,2,6,...42,46,48}`

[1]For grid4_1.mat each item listed, aside from source node κ_s and target node κ_t, the edge-weight default value for all index positions is set to 1, i.e., $\lambda_\eta = 1$.

Table D.7
Energy management model: demand_intialise.mat

Item	Name	Value	Parameter
1	coeff	$13 \times 4 \times 2$ double	See Chapter 4 Table 4.2
2	$cs365_{max}$	`123.1363`	-
3	$cs365_{min}$	`-10.3966`	-
4	horizon	`0`	-
5	monthLUT	1×12 double	See Chapter 4 Table 4.1

Appendix E

Case Study: Software Code

Appendix Contents

E.1 Energy management model: signal inspector

Table E.1 lists all signals defined during the verification workflows. Line styles and colours are kept consistent for every simulation run.

Table E.1
Energy management model: signal inspector

Signal Name	Colour	RGB	Location
temp_SP	dark grey	(51,51,0)	L0 - main
SOC	blue	(0,102,204)	L0 - main
DIR	orange	(255,153,0)	L0 - main
FIT	black	(0,0,0)	L0 - main
PWR	red	(255,0,0)	L0 - main
t_mode	green	(0,127,0)	L0 - main
temp_room	olive	(204,204,0)	building_subsystem
temp_out	red	(162,20,47)	building_subsystem
cost	teal	(153,153,0)	building_subsystem
tariff	blue	(0,14,255)	building_subsystem
des_mode	magenta	(255,0,255)	des_subsystem
des_end	red	(255,0,0)	des_subsystem
des_begin	green	(0,133,0)	des_subsystem
occupants	blue	(0,102,204)	dt_subsystem
response	green	(0,153,0)	dt_subsystem
comfort	red	(153,0,0)	dt_subsystem
demand	blue	(0,112,112)	dt_subsystem
Δf	grey	(128,128,128)	energy_subsystem
t_mode*	magenta	(255,0,255)	scheduler_subsystem
tou_tariff*	orange	(255,153,0)	scheduler_subsystem
demand	blue	(0,112,112)	gridmap: DEMAND
rescale	red	(192,0,0)	gridmap: DEMAND
spline	green	(76,153,0)	gridmap: DEMAND
occupants	blue	(0,102,204)	gridmap: COMFORT
response	green	(0,153,0)	gridmap: COMFORT
comfort	red	(153,0,0)	gridmap: COMFORT
tou	magenta	(255,0,255)	gridmap: TOU
comfort	red	(153,0,0)	gridmap: GROUP
demand	blue	(0,112,112)	gridmap: GROUP
tou	magenta	(255,0,255)	gridmap: GROUP
forecast	green	(34,139,34)	gridmap: GROUP

E.2 hil_optim_ctrl.m

```
function optim_ctrl(block)
%% OPTIM_CTRL *HIL
%
%  Title: Optimisation and Control (HIL)
%  Filename: optim_ctrl.m
%  Prepared by: Sean Williams
%  Date: 20 Dec 2019
%
%  Code tagged to Simulink model block optimise_subsystem. On receipt of
%  date/time (sampe rate: 10 minute) code computes new temperature
%  setpoint (ctrl_action) using Dijsktra's algorithm which is a function
%  of occupant thermal comfort, electricity demand and cost (tariff).
%  Code reacts on receipt of demand event signal
%
%  > HIL
%  1. New input port InputPort(2).Data for S1tcv
%  2. S1tcv variable is passed to function prepare_tc_gridmap
%     Case 1 Section 3
%

setup(block);

%endfunction: optim_ctrl(block)

function setup(block)
%% Setup Functional Port Properties

% Register number of ports
block.NumInputPorts  = 4;
block.NumOutputPorts = 3;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Override input port properties
block.InputPort(1).Dimensions  = 1;  % temp_room
block.InputPort(1).DatatypeID  = 0;  % double
block.InputPort(1).Complexity  = 'Real';
block.InputPort(1).DirectFeedthrough = true;

% Override input port properties
block.InputPort(2).Dimensions  = 1;  % S0tcv
block.InputPort(2).DatatypeID  = 0;  % double
block.InputPort(2).Complexity  = 'Real';
block.InputPort(2).DirectFeedthrough = true;

% Override input port properties
block.InputPort(3).Dimensions  = 1;  % S0_date
block.InputPort(3).DatatypeID  = 0;  % double
block.InputPort(3).Complexity  = 'Real';
block.InputPort(3).DirectFeedthrough = true;

% Override input port properties
block.InputPort(4).Dimensions  = 1;  % des_mode
block.InputPort(4).DatatypeID  = 0;  % double
block.InputPort(4).Complexity  = 'Real';
block.InputPort(4).DirectFeedthrough = true;


% Override output port properties
block.OutputPort(1).Dimensions  = 1; % ctrl_action
block.OutputPort(1).DatatypeID  = 0; % double
block.OutputPort(1).Complexity  = 'Real';

% Override output port properties
block.OutputPort(2).Dimensions  = 1; % tou_tariff
block.OutputPort(2).DatatypeID  = 0; % double
block.OutputPort(2).Complexity  = 'Real';

% Override output port properties
block.OutputPort(3).Dimensions  = 1; % des_duration
block.OutputPort(3).DatatypeID  = 0; % double
block.OutputPort(3).Complexity  = 'Real';

% Register parameters
```

```
77  block.NumDialogPrms = 0;
78
79  % Register sample times
80  block.SampleTimes = [600 0];
81
82  % Specify the block simStateCompliance to default
83  block.SimStateCompliance = 'DefaultSimState';
84
85  % Register nethods
86  block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
87  block.RegBlockMethod('InitializeConditions', @InitializeConditions);
88  block.RegBlockMethod('Start', @Start);
89  block.RegBlockMethod('Outputs', @Outputs); % Required
90  block.RegBlockMethod('Update', @Update);
91  block.RegBlockMethod('Derivatives', @Derivatives);
92  block.RegBlockMethod('Terminate', @Terminate); % Required
93  block.RegBlockMethod('SetInputPortSamplingMode',@SetInpPortFrameData);
94
95  %endfunction: setup(block)
96
97  function DoPostPropSetup(block)
98
99  % Initialise the Dwork vectors
100 block.NumDworks = 4;
101
102 % Dwork(1) stores the status of the count_flag [count_flag]
103 block.Dwork(1).Name            = 'D1';
104 block.Dwork(1).Dimensions      = 1;
105 block.Dwork(1).DatatypeID      = 0;      % double
106 block.Dwork(1).Complexity      = 'Real'; % real
107 block.Dwork(1).UsedAsDiscState = true;
108
109 % Dwork(2) stores the value of the counter [count]
110 block.Dwork(2).Name            = 'D2';
111 block.Dwork(2).Dimensions      = 1;
112 block.Dwork(2).DatatypeID      = 0;      % double
113 block.Dwork(2).Complexity      = 'Real'; % real
114 block.Dwork(2).UsedAsDiscState = true;
115
116 % Dwork(3) stores the nodepath as a vector when a dv event is initiated [dv_event]
117 block.Dwork(3).Name            = 'D3';
118 block.Dwork(3).Dimensions      = 25;
119 block.Dwork(3).DatatypeID      = 0;      % double
120 block.Dwork(3).Complexity      = 'Real'; % real
121 block.Dwork(3).UsedAsDiscState = true;
122
123 % Dwork(4) stores the status of the des_end [des_end]
124 block.Dwork(4).Name            = 'D4'; %des_end
125 block.Dwork(4).Dimensions      = 1;
126 block.Dwork(4).DatatypeID      = 0;      % double
127 block.Dwork(4).Complexity      = 'Real'; % real
128 block.Dwork(4).UsedAsDiscState = true;
129
130 %endfunction: DoPostPropSetup(block)
131
132 function InitializeConditions(block)
133 %% Set Initial Conditions
134
135 % Set the initial status of the count_flag to zero
136 block.Dwork(1).Data=1;
137
138 % Set the initial value of demand event counter to 24
139 % 24 is baseline counter required for 4 hour ramp time, and before event
140 % duration counter is applied
141 block.Dwork(2).Data=24;
142
143 % Set initial status of des_end to zero
144 % des_end=0 [no demand event signal]
145 block.Dwork(4).Data=0;
146
147 % Set Simulink Model block parameters
148 set_param('optim_ctrl_model_sim/des_subsystem/des_end','Value','0')
149
150 % Set des_duration output signal to zero
151 block.OutputPort(3).Data=0;
152
153 %endfunction: InitializeConditions(block)
154
155 function Start(block)
156 %% Set Start Conditions
157
158 % Assign Dwork(1) to status of count_flag
```

```
159  block.Dwork(1).Data=1;
160
161  % Assign Dwork(4) status of des_end
162  block.Dwork(4).Data=block.InputPort(4).Data;
163
164  %endfunction: Start(block)
165
166  function Outputs(block)
167  %% Outputs
168
169  % Set Simulink block parameters
170  temperature=block.InputPort(1).Data;
171  S1tcv=block.InputPort(2).Data;
172  des_end=block.Dwork(4).Data;
173  count_flag=block.Dwork(1).Data;
174  count=block.Dwork(2).Data;
175  dv_event=block.Dwork(3).Data;
176  set_param('optim_ctrl_model_sim/des_subsystem/des_end',...
177      'Value','0'); % reset des_end at end of event
178
179  % set S0_date to 'base' for initial cycle only, then revert to date on Input Port 4
180  if block.InputPort(3).Data<10
181      S0_date=datetime(datestr(evalin('base','dt')));
182  else
183      S0_date=datetime(datestr(block.InputPort(3).Data));
184  end
185
186  % Initialise model
187  [visual_mode,horizon,gridmap,edgepath,t1,temp_step,event_duration]=initialise(S0_date);
188
189  % Set edgepath for each function
190  %       [1] comfort
191  %       [2] demand
192  %       [3] tou (tariff)
193  %       [4] optim (ALL)
194  for edgepage=1:4
195
196      switch edgepage
197
198          case 1 % comfort
199
200              % 1. INITIALISE (LOCAL)
201              % =====================
202              % set local paramaters
203
204              % set minimum temperature threshold
205              % if nil occupancy edge weight will force path to reduce temperature
206              % setpoint until minimum temperature threshold is reached. At which point
207              % edge weight will force maintain minimum temperature threshold
208              mintempthreshold=16;
209
210              % set minimum temperature threshold parameter
211              % translate minimum temperature threshold to equivalent grid map value
212              mintemp=[17.5 17 16.5 16 15.5];
213              mintempvar=[17 20 23 26 29];
214              mintempthresholdparam=mintempvar(mintemp==mintempthreshold);
215
216              % define array that describes range of temperatures (nodes) at time t0
217              t0tempSP=[20.5 20 19.5 19 18.5 18 17.5 17 16.5 16 15.5];
218              % t0nodes=1:1:11;
219
220              % 2a. PREPARE COMFORT VALUES
221              % ==========================
222              [tempSP]=prepare_comfort_values(temperature);
223
224              % 3. PREPARE GRIDMAP
225              % ==================
226              [tcv,tcvdata,gridmap_c,t0minidx,t240minidx]=prepare_tc_gridmap(t0tempSP,tempSP,...
227                  S0_date,gridmap,edgepath,edgepage,mintempthresholdparam,S1tcv);
228
229              % 4. PREPARE DIGRAPH
230              % ==================
231              [G_c,B]=prepare_digraph(gridmap_c,edgepage);
232
233              % 5. DIJKSTRA
234              % ===========
235              [~,path_c]=dijkstra(B,gridmap_c(t0minidx,1,edgepage),gridmap_c(t240minidx,71,...
236                  edgepage));
237              [edgepath_c]=prepare_edgepath(gridmap_c,edgepage,path_c);
238
239              % 6. VISUALISATION: COMFORT (4H/24H HORIZON WINDOW)
```

```
240             % ================================================
241             % fixed subplot showing scrolling comfort [1,4,1]
242             if (visual_mode==4)
243                 visual_comfort_data(horizon,tcvdata,t1);
244             end
245
246             % 7. VISUALISATION: GRIDMAP INDIVIDUAL SHORTESTPATH
247             % ================================================
248             % new figure created for each cycle showing individual gridmap
249             %fig_name='Comfort Gridmap';
250             %edgepath_color=[153/255 0 0]; % red
251             %visual_individual_shortestpath(fig_name,G_c,edgepage,edgepath_c,t1,edgepath_color);
252
253         case 2 % demand
254
255             % 1. INITIALISE (LOCAL)
256             % =====================
257
258
259             % 2a. PREPARE DEMAND VALUES
260             % =========================
261             [dv,dv_gridmap,dv_nodepath,dv_rescale]=prepare_dv_values(horizon,S0_date);
262
263             % 2b. PREPARE NODE PATH
264             % =====================
265                 if(des_end==1)
266                     % on receipt of demand event signal set grid path to increase by 2degC or
267                     % 3degC [temp_step] from last recorded temperature over 4hr horizon window at
268                     % increments of 0.5degC every 50 min [2degC] or 40 min [3degC].
269                     if (count_flag==1)
270                         block.OutputPort(3).Data=0;
271
272                         count_flag=0; % set flag to false to ensure loop is executed only once
273                         block.Dwork(1).Data=count_flag;
274                         % set node start point (source) temperature
275                         node=dv_rescale(1,:);
276                         %node=10 % test
277
278                         dv_nodepath=[]; % temporary placeholder for new path
279
280                         switch temp_step
281                             case 2
282                             dv_nodepath=zeros(25,1); % initialise variable dv_nodepath
283                             % set path for 2degC increase over 4hr period increasing 0.5degC at
284                             % 50min intervals
285                                 for p1=0:5:20
286                                     for p2=1:5
287                                         if node-(p1/5)<1
288                                             dv_nodepath(p2+p1,1)=1;
289                                         else
290                                             dv_nodepath(p2+p1,1)=node-(p1/5);
291                                         end
292                                     end
293                                 end
294                             case 3
295                             dv_nodepath=zeros(25,1); % initialise variable dv_nodepath
296                             % set path for 3degC increase over 4hr period increasing 0.5degC at
297                             % approximately 40min intervals
298                                 for p1=0:4:25
299                                     for p2=1:4
300                                         if node-(p1/4)<1
301                                             dv_nodepath(p2+p1,1)=1;
302                                         else
303                                             dv_nodepath(p2+p1,1)=node-(p1/4);
304                                         end
305                                     end
306                                 end
307                                 % trim dv_nodepath
308                                 dv_nodepath(1,:)=[];
309                                 dv_nodepath(26:end,:)=[];
310                         end
311                         dv_event=dv_nodepath;
312                         block.Dwork(3).Data=dv_event;
313
314                     % on receipt of demand event signal, and after inital path showing
315                     % increase of either 2degC or 3degC [temp_step] code begins to scroll
316                     % grid map horizontally by one-step at each 10 minute cycle.
317                     else
318                         if (count>1)
319                             dv_nodepath(1:count-2,1)=dv_event(size(dv_nodepath,1)-count+2: ...
320                                 end-1,:);
```

```
                            if ((dv_event(1,1)+temp_step)>11)
                                dv_nodepath(count-1:count+event_duration,:)=11;
                            else
                                dv_nodepath(count-1:count+event_duration,:)=dv_event(1)+2;
                            end
                        else
                            if ((dv_event(1,1)+temp_step)>11)
                                dv_nodepath(1:count+event_duration,:)=11;
                            else
                                dv_nodepath(1:count+event_duration,:)=dv_event(1)+2;
                            end
                        end
                        count=count-1;
                        block.Dwork(2).Data=count;

                        % this will trigger use of ESS for duration of
                        % demand side event, try this first then remember
                        % to reset new output back to zero in the next if
                        % loop below
                        if count==1
                            block.OutputPort(3).Data=3;
                        end

                        if (count==-event_duration-1)
                            set_param('optim_ctrl_model_sim/des_subsystem/des_end',...
                                'Value','1');
                            count_flag=1;
                            block.Dwork(1).Data=count_flag;
                            count=24; % reset count
                            block.Dwork(2).Data=count;
                            block.Dwork(4).Data=0;
                        end
                    end
                end

            % 3. PREPARE GRIDMAP
            % ==================
            [t0minidx,t240minidx,gridmap_d]=prepare_gridmap(dv_nodepath,gridmap,edgepage,...
                edgepath);

            % 4. PREPARE DIGRAPH
            % ==================
            [G_d,B]=prepare_digraph(gridmap_d,edgepage);

            % 5. DIJKSTRA
            % ===========
            [~,path_d]=dijkstra(B,gridmap_d(t0minidx,1,edgepage),gridmap_d(t240minidx,71,...
                edgepage));
            [edgepath_d]=prepare_edgepath(gridmap_d,edgepage,path_d);

            % 6. VISUALISATION: Demand (4H/24H HORIZON WINDOW)
            % ================================================
            % fixed subplot showing scrolling demand [1,4,2]
            if (visual_mode==3) || (visual_mode==4)
                visual_demand_data(horizon,dv,dv_gridmap,dv_rescale,t1)
            end

            % 7. VISUALISATION: GRIDMAP INIDVIDUAL SHORTESTPATH
            % =================================================
            % new figure created for each cycle showing individual gridmap
            %fig_name='Demand Gridmap';
            %edgepath_color=[0 112/255 192/255]; % blue
            %visual_individual_shortestpath(fig_name,G_d,edgepage,edgepath_d,t1,edgepath_color)

        case 3 % tou

            % 1. INITIALISE (LOCAL)
            % =====================
            % set local parameters
            tou_tariff=[0.0499 0.1199 0.2499];
            period={'00:00:00','06:00:00','16:00:00','19:00:00','23:00:00','24:00:00'};

            % 2a. PREPARE TOU VALUES
            % ======================
            [touv_gridmap,touv_nodepath,touv_rescale,touv]=prepare_tou_values(horizon,...
                S0_date,period,tou_tariff);

            touv_op=touv(1,1);

            % 3. PREPARE GRIDMAP
            % ==================
```

```
402             [t0minidx,t240minidx,gridmap_t]=prepare_gridmap(touv_nodepath,gridmap,edgepage,...
403                 edgepath);
404             %gridmap_t=gridmap;
405
406             % 4. PREPARE DIGRAPH
407             % ==================
408             [G_t,B]=prepare_digraph(gridmap_t,edgepage);
409
410             % 5. DIJKSTRA
411             % ===========
412             [~,path_t]=dijkstra(B,gridmap_t(t0minidx,1,edgepage),gridmap_t(t240minidx,71,...
413                 edgepage));
414             [edgepath_t]=prepare_edgepath(gridmap_t,edgepage,path_t);
415
416             % 6. VISUALISATION: TOU (4H/24H HORIZON WINDOW)
417             % =============================================
418             % fixed subplot showing scrolling tou [1,4,3]
419             if (visual_mode==4)
420                 visual_tou_data(horizon,touv_rescale,t1)
421             end
422
423             % 7. VISUALISATION: GRIDMAP INDIVIDUAL SHORTESTPATH
424             % =================================================
425             % new figure created for each cycle showing individual gridmap
426             %fig_name='TOU Gridmap';
427             %edgepath_color=[204/255 0 153/255]; % magenta
428             %visual_individual_shortestpath(fig_name,G_t,edgepage,edgepath_t,t1,edgepath_color)
429
430         case 4 % optim
431
432             % 1. INITIALISE (LOCAL)
433             % =====================
434             % set local parameters
435             stage_centroid=1;
436             X=zeros(3,2);
437
438             % 2. PREPARE VALUES
439             % =================
440             % not required
441
442             % 3. PREPARE GRIDMAP
443             % ==================
444             % set gridmap to multidimensional array template (31x72x4 double)
445             gridmap(:,:,1)=gridmap_c(:,:,1);
446             gridmap(:,:,2)=gridmap_d(:,:,2);
447             gridmap(:,:,3)=gridmap_t(:,:,3);
448             gridmap(:,:,4)=gridmap_c(:,:,4);
449
450             for s=3:3:72
451                 for p=1:(edgepage-1)
452                     X(p,1)=find(gridmap(:,s,p)==min(gridmap(:,s,p)));
453                     X(p,2)=0;
454                     [~,c]=kmeans(X,1);
455                     id=floor(c(1));
456                 end
457
458                 gridmap(id,s,edgepage)=stage_centroid;
459                 for j=id-1:-1:1
460                     gridmap(j,s,edgepage)=stage_centroid+id-j;
461                 end
462                 for j=id+1:1:31
463                     gridmap(j,s,edgepage)=stage_centroid+j-id;
464                 end
465             end
466
467             t0minidx=find(gridmap(:,3,4)==min(gridmap(:,3,4)));
468             t240minidx=find(gridmap(:,72,4)==min(gridmap(:,72,4)));
469
470             % 4. PREPARE DIGRAPH
471             % ==================
472             [G_a,B]=prepare_digraph(gridmap,edgepage);
473
474             % 5. DIJKSTRA
475             % ===========
476             [~,path_a]=dijkstra(B,gridmap(t0minidx,1,edgepage),gridmap(t240minidx,71,edgepage));
477             [edgepath_a]=prepare_edgepath(gridmap,edgepage,path_a);
478
479             % 6. VISUALISATION: DATA (4H/24H HORIZON WINDOW)
480             % ==============================================
481             % not required
482
```

```
483             % 7. VISUALISATION: GRIDMAP INDIVIDUAL SHORTESTPATH
484             % ================================================
485             % not required
486
487             % 8. CONTROL ACTION
488             % ==================
489             % control action is temperature at t10, S1
490              ctrl_stage=2;
491              ctrl_action=t0tempSP(path_a(ctrl_stage)-(11*(ctrl_stage-1)));
492
493             % 9. VISUALISATION: BIG PATH
494             % ==========================
495             %fixed subplot showing scrolling total cost function [1,4,4]
496              if (visual_mode==2) || (visual_mode==3) || (visual_mode==4)
497                  visual_group_path(path_c, path_d, path_t, path_a, t1)
498              end
499
500             % 10. VISUALISATION: BIG GRIDMAP SHORTESTPATH
501             % ===========================================
502             % new figure created for each cycle showing individual gridmap
503             %visual_group_shortestpath(G_c,edgepath_c,G_d,edgepath_d,G_t,edgepath_t,G_a,...
504             %edgepath_a,t1,horizon)
505         end
506     end
507
508     % Update Simulink model output ports
509     block.OutputPort(1).Data = ctrl_action ;
510     block.OutputPort(2).Data = touv_op;
511
512
513     %endfunction: Outputs(block)
514
515     function Update(block)
516     %% Update Dwork
517
518     % Update Dwork(4) to InputPort(4) [S0_date]
519     block.Dwork(4).Data=block.InputPort(4).Data;
520
521     %endfunction: Update(block)
522
523     function SetInpPortFrameData(block, idx, fd)
524
525     % Set the sampling of the input ports
526     block.InputPort(idx).SamplingMode=fd;
527     for i=1:block.NumOutputPorts
528         block.OutputPort(i).SamplingMode=fd;
529     end
530
531     %endfunction: SetInpPortFrameData(block,idx,fd)
532
533     function Terminate(block)
534
535     %endfunction: Terminate(block)
```

Listing E.1 hil_optim_ctrl.m

E.3 hil_optim_ctrl_model_data.m

```
%% MATLAB M-File Description *HIL
%
%  Title: Optimisation and Control Model Building Parameters (HIL)
%  Filename: optim_ctrl_model_data.m
%  Prepared by: Sean Williams
%  Date: 24 Oct 2019
%
%  MATLAB script tagged to Simulink model 'optim_ctrl_model_sim.slx',
%
%  >HIL
%  1. The following sections have been deleted:
%       Set Outdoor Temperature Variation
%       Set Building Paremeters
%       Set Power Systems Parameters
%  2. Modified Set Date Time: delete references to daily_temp include
%  ifelse statement at end of section. Case 1-6 date_time remains
%  unchanged.
%

%% Change History
%
%   1. [24-10-2019] Initial
%   2. [20-12-2019] New: Set Initialise Parameters, Set Date Time,
%   (option to use dtv_sim or dtv_act), Set SOC Model Parameters,
%   Set Outdoor Temperature Variation; Modified: Set Building Parameters
%   3. [26-01-2020] New: energy_subsystem parameters
%   4. [17-03-2020] Changes required to support HIL
%   5.

%% Set Initialise Parameters

Δ1=1.157412771135569e-05;
Δ10=0.006944444445185;

%% Set Date Time

% Option to select simulated daily temperature variation [1|2]
% or measured daily temperature [3|4|5|6]
% https://www.wunderground.com <act_temp.xlsx>

date_time_option=1;

switch date_time_option
    case 1
        date_time='now';

    case 2
        date_time='10:00:00';

    case 3 % Sunday 10-Feb-2019 00:00:00, 24hrs
        date_time='10-Feb-2019 00:00:00'; %datenum=737466

    case 4 % Tuesday 07-May-2019 00:00:00, 24hrs
        date_time='7-May-2019 00:00:00'; %datenum=737552

    case 5 % Saturday 3-Aug-2019 00:00:00, 24hrs
        date_time='3-Aug-2019 00:00:00'; %datenum=737640

    case 6 % Friday 08-Nov-2019 00:00:00, 24hrs
        date_time='8-Nov-2019 00:00:00'; %datenum=737737

end

dt=datenum(datetime(date_time));
assignin('base','dt',dt);


%% Set SOC Model Parameters

SDR=0.017;
tau=2000;
tou_weekday=[4.99 11.99 24.99 11.99 4.99];
```

Listing E.2 hil_optim_ctrl_model_data.m

E.4 hil_prepare_tc_gridmap.m

```
function [tcv,tcvdata,gridmap,t0minidx,t240minidx]=prepare_tc_gridmap ...
    (t0tempSP,tempSP,S0_date,gridmap,edgepath,edgepage, ...
    mintempthresholdparam,S1tcv)
%% MATLAB Function Description *HIL
%
%  Title: Prepare Thermal Comfort Gridmap (HIL)
%  Filename: prepare_tc_gridmap.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function starts with gridmap (31x72x4) template. Maps nodepath
%  (11x25) onto gridmap (31x72) for each objective function (page):
%  comort, demand and tou (tariff).
%  At each stage the min value is defined as the stage centroid,
%  all remaining values are populated, increasing/decreasing in
%  value moving up/down in the same col (stage). The index where the
%  min value at t0 and t240 is found are stored in t0minidx and t240minidx
%  respectively.
%  Exception handling at boundary upper and lower is included.
%
%  Similar to prepare_gridmap.m but specific to thermal comfort.
%
%  > HIL
%  General: code change enables signal from single smart phone to interact
%  with Simulink model. tc at S1 is set to feedback from smart phone irrespective
%  of planned occupancy. Code that sets response for S2 to S24 remains unchanged.
%  1. Include S1tcv as input parameter
%  2. tcv(3) (calc_mode) at S1 set to S1tcv (user themal comfort feedback)
%

%% Change History
%
%    1. [06-11-2019] Initial
%    2. [15-03-2020] Changes required to support HIL
%    3.

%% Prepare Thermal Comfort Gridmap

% Define stage S1 column number used in grid template, multiples of col is
% used to calculate S2 to S24
col=3;

% Find index number of value in array t0tempSP that matches the recorded
% temperature measurement
node=find(t0tempSP==tempSP);

% Calculate node index of declared temperature setpoint at t0; (2016≤nodeidx=<1)
nodeidx=(node*2)+(node-2);

% tc at S1
% ========
% Calculate thermal comfort value (tcv) edge
% tcv=comfort(stime,1) where stime=[10,20,...,n] minutes
% no difference if users thermal comfort calc_mean is COLD or TOO COLD.
% Also no difference to control action if users thermal comfort
% calc_mean is WARM or TOO WARM.
% Returns [n1 n2 n3] where n1=occupants, n2=response, n3=calc_mode
Sn_date=datetime(S0_date,'ConvertFrom','datenum');
tcv=comfort_2(Sn_date,1);

% HIL, force tcv(3) (calc_mode) to S1tcv (thermal comfort user feedback)
tcv(3)=S1tcv;

tcvdata(1,:)=tcv;

% Set initial edge weight (S1)
% Check number of op. If op=0 then set edge weight to direct path
% to reduce tempSP by 0.5degC. Otherwise set edge weight based on
% return fcn: comfort value.
% If nil op then set path to reduce tempSP by 0.5degC from S0 to S1
if (tcv(1)==0)
    a=1;b=0.5;c=0.25;
else
    % tc=-1 or -2 indicating too hot, reduce tempSP. Least value path
    % is to lower temp
    if (tcv(3)<0)
```

```
77              a=1;b=0.5;c=0.25;
78      % tc=0 indicating okay, maintain tempSP. Least value path is to
79      % same temp
80       elseif (tcv(3)==0)
81              a=0.5;b=0.25;c=0.5;
82      % tc=1 or 2 indicating too cold, increase tempSP, Least value path to
83      % higher temp.
84       elseif (tcv(3)>0)
85              a=0.25;b=0.5;c=1.0;
86      end
87  end
88
89  % Check for outliner temperature values (20.5 and 15.5) and restrict setting
90  % of edge weights to valid edges, ie not possible to assign edge weight
91  % from 20.5 to 21.0.
92  % higher sn<tn (source node is less than targe node)
93   if (nodeidx-1==0)
94       gridmap(nodeidx,col,edgepage)=b;
95       gridmap(nodeidx+1,col,edgepage)=c;
96  % lower sn>tn
97   elseif (nodeidx-31==0)
98       gridmap(nodeidx-1,col,edgepage)=a;
99       gridmap(nodeidx,col,edgepage)=b;
100 % equal sn=tn
101  else
102      gridmap(nodeidx-1,col,edgepage)=a;
103      gridmap(nodeidx,col,edgepage)=b;
104      gridmap(nodeidx+1,col,edgepage)=c;
105  end
106
107 % Find row number (t0minidx) listing minimum edge weight at S1 use t0minidx
108 % in dijkstra.mlx to set source node when calculating shortest path
109 [t0value,t0minidx]=min(gridmap(:,3,edgepage));
110
111 % tc at S2 to S24
112 % ===============
113  for n=1:23 %23
114      % Display label > 'Stage: n (k)' where n=[2,3,...,24], k=[6,9,...,72]
115      % column number disp(['Stage: ',num2str(n+1),' (',num2str((n*3)+3),') '])
116
117      Sn_date=datetime(S0_date,'ConvertFrom','datenum')+minutes(10*n);
118
119      % Calculate tc for next stage
120      tcv=comfort_2(Sn_date,0);
121      tcvdata(n+1,:)=tcv;
122
123      % Determine value (not required) and idx of min value from previous stage
124      [¬,idx]=min(gridmap(:,col*n,edgepage));
125
126      % Determine the previous stage target node
127      tnode=gridmap(idx,(col*n)-1,edgepage);
128
129      % Now declare the start node of next stage the target node from
130      % previous stage
131      snode=tnode;
132
133      % Determine the node index number of min value in previous stage
134      nodeidx=sub2ind(size(gridmap(:,:,edgepage)),idx,col*n);
135
136      % Find the index numbers of the start nodes in the next stage
137      % (with exception to outliners, there are three values returned)
138      [value,¬]=find(gridmap(:,1+(col*n),edgepage)==snode);
139
140      % Maintain list of edgepath of shortest path in grid map
141      edgepath=cat(2,edgepath,sub2ind(size(gridmap(:,:,edgepage)),idx,n));
142
143      % Check if op>0
144      if(tcv(1)>0)
145          % This ensures 1:12 maintains 12:23, 23:34 etc
146          if (value(1)==1)
147              gridmap(value(1),3+(col*n),edgepage)=t0value;
148          else
149              gridmap(value(2),3+(col*n),edgepage)=t0value;
150          end
151      else
152          % This ensures 1:n starts to fall when no occupancy
153          if (value(1)==1)
154              gridmap(value(1)+1,3+(col*n),edgepage)=t0value;
155          else
156              % Check path exceeds declared minimum temperature threshold value
157              if (idx<mintempthresholdparam)
```

```
158                 % Continue to force shortest path to lower temperature
159                 % if value is less than declared minimum temperature
160                 % threshold value
161                 gridmap(value(2)+1,3+(col*n),edgepage)=t0value;
162             else
163                 % Maintain shortest path on minimum temperate threshold
164                 % value if calculated temperature is less than minimum
165                 % temperature threshold value
166                 gridmap(value(2),3+(col*n),edgepage)=t0value;
167             end
168         end
169     end
170 end
171
172 % Compute row number showing lowest value at S24
173 % values are used when plotting shortest path
174 [~,t240minidx]=min(gridmap(:,72,edgepage));
175
176 % Add final node to edgepath list that informs shortest path
177 edgepath(1,end+1)=(23*31)+t240minidx;
178
179 %end prepare tc gridmap
```

Listing E.3 hil_prepare_tc_gridmap.m

E.5 hil_read_serialdata.m

```
function [t0, S1tcv] = read_serialdata
%% MATLAB Function Description *HIL
%
%  Title: Read Serial Data (HIL)
%  Filename: read_serialdata.m
%  Prepared by: Sean Williams
%  Date: 15 Mar 2020
%
%  MATLAB function reads room temperature and thermal comfort from serial
%  port. COM port connects to Industruino using USB.
%

%% Change History
%
%   1. [15-03-2020] Initial
%   2.

%% Read Serial Data
% set port parameters
s=serial('COM16'); % USB to Industruino
set(s,'BaudRate',9600);
set(s,'DataBits',8);
set(s,'StopBits',1);
set(s,'Parity','none');
set(s,'Timeout',8);
set(s,'Terminator','CR/LF');

% disable waning message
warning('off','MATLAB:serial:fscanf:unsuccessfulRead')

% open port
fopen(s)
s.ReadAsyncMode='continuous';
%lastwarn('');
[a, MSGID] = lastwarn('');
try
    % read data <ff.ff;d;CR/LN>
    % where ff.ff is room temperature in degC
    % B is thermal comfort (ascii in {48,49,50,51,52})
    C=fscanf(s,'%f %*c %d %*c',9)

    if(~isempty(lastwarn))
        error(lastwarn)
    end
catch err
   C=[99;48];
end

% close port
fclose(s)
stopasync(s)

% enable warning message
warning on verbose

% set temperature and thermal comfort parameters
t0=C(1);
S1tcv=str2num(char(C(2)));

% set t0=99 if fail read
% (strlength(num2str(t0))≠4) ||
if (~isnumeric(t0)) ||  ...
        (~isfloat(t0)) || (size(t0,1)>1) || ...
        (size(t0,2)>1)
    t0=99;
end

%set S1tcv=0 is fail read
if (isempty(S1tcv)) || (~isnumeric(S1tcv)) || ...
        (S1tcv<0) || (S1tcv>4)
    S1tcv=0;
end

end
```

```
75  % end read_serialdata
```

Listing E.4 hil_read_serialdata.m

E.6 hil_readdata.m

```
function readdata(block)
%% READDATA *HIL
%
%  Title: Read Data (HIL)
%  Filename: readdata.m
%  Prepared by: Sean Williams
%  Date: 9 Mar 2020
%
%  Read room temperature and thermal comfort values from serial port.
%  Both data routed from Industruino using USB connected to serial port
%  with room temperature from remote Arduino Uno and DHT22 sensor using
%  wireles connection and thermal comfort from Android smart phone
%  using bluetooth connection.
%
%  Function outputs include room temperature plus 2 further options:
%       1. room temperature -2 degC
%       2. room temperature -4 degC
%  Final output is thermal comfort
%
%  Data exception handling included
%  Sample rate is 5min (300s)
%

 setup(block);

%endfunction: readdata(block)

 function setup(block)
%% Setup Functional Port Properties

% Register number of ports
 block.NumInputPorts  = 0;
 block.NumOutputPorts = 4;

% Setup port properties to be inherited or dynamic

 block.SetPreCompOutPortInfoToDynamic;

% Override output port properties
 block.OutputPort(1).Dimensions  = 1; % t0: temp
 block.OutputPort(1).DatatypeID  = 0; % double
 block.OutputPort(1).Complexity  = 'Real';
 block.OutputPort(1).SamplingMode='Sample';

% Override output port properties
 block.OutputPort(2).Dimensions  = 1; % t1: temp-2
 block.OutputPort(2).DatatypeID  = 0; % double
 block.OutputPort(2).Complexity  = 'Real';
 block.OutputPort(2).SamplingMode='Sample';

% Override output port properties
 block.OutputPort(3).Dimensions  = 1; % t2: temp-3
 block.OutputPort(3).DatatypeID  = 0; % double
 block.OutputPort(3).Complexity  = 'Real';
 block.OutputPort(3).SamplingMode='Sample';

% Override output port properties
 block.OutputPort(4).Dimensions  = 1; % S1tcv: thermal comfort at Stage 1
 block.OutputPort(4).DatatypeID  = 0; % double
 block.OutputPort(4).Complexity  = 'Real';
 block.OutputPort(4).SamplingMode='Sample';

% Register parameters
 block.NumDialogPrms = 0;

% Register sample times
 block.SampleTimes = [300  0];

% Specify the block simStateCompliance to default
 block.SimStateCompliance = 'DefaultSimState';

% Register nethods
 block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
 block.RegBlockMethod('InitializeConditions', @InitializeConditions);
 block.RegBlockMethod('Start', @Start);
 block.RegBlockMethod('Outputs', @Outputs); % Required
```

```
77  block.RegBlockMethod('Update', @Update);
78  block.RegBlockMethod('Terminate', @Terminate); % Required
79  block.RegBlockMethod('SetInputPortSamplingMode',@SetInpPortFrameData);
80
81  %endfunction: setup(block)
82
83  function DoPostPropSetup(block)
84
85  % Initialise the Dwork vectors
86  block.NumDworks = 2;
87
88  % Dwork(1) stores the value of the last room temperature reading
89  block.Dwork(1).Name            = 'T1';
90  block.Dwork(1).Dimensions      = 1;
91  block.Dwork(1).DatatypeID      = 0;      % double
92  block.Dwork(1).Complexity      = 'Real'; % real
93  block.Dwork(1).UsedAsDiscState = true;
94
95  % Dwork(2) stores the value of the last thermal comfort value
96  block.Dwork(2).Name            = 'S1tcv';
97  block.Dwork(2).Dimensions      = 1;
98  block.Dwork(2).DatatypeID      = 0;      % double
99  block.Dwork(2).Complexity      = 'Real'; % real
100 block.Dwork(2).UsedAsDiscState = true;
101
102 %endfunction: DoPostPropSetup(block)
103
104 function InitializeConditions(block)
105 block.Dwork(1).Data=18;
106 block.Dwork(2).Data=0;
107 %endfunction: InitializeConditions(block)
108
109 function Start(block)
110 %% Set Start Conditions
111
112 % Assign Dwork(1) to status of last temp
113 block.Dwork(1).Data=18;
114 block.Dwork(2).Data=0;
115 %endfunction: Start(block)
116
117 function Outputs(block)
118 %% Outputs
119 % hold previous temperature value
120 lastt0=block.Dwork(1).Data;
121
122 % hold previous thermal comfort value
123 lastS1tcv=block.Dwork(2).Data;
124
125 % read new serial data
126 [t0,S1tcv]=read_serialdata
127
128 % if temperature data error set new temperature to previous temperature
129 if (t0 == 99)
130     t0=lastt0;
131 end
132
133 % if thermal comfort data error set new thermal comfort data to '0' (user
134 % reports thermal comfort is 'warm enough')
135 if (S1tcv == 99)
136     S1tcv=0;
137 end
138
139 % map thermal comfort serial in to tcalc_mode
140 %   thermal comfort    serial in    tcalc_mode
141 %   it's warm enough   0             0
142 %   its cold           1             1
143 %   it's too cold      2             2
144 %   it's warm          3            -1
145 %   it's too warm      4            -2
146 tcalc_mode_mapping = [0,1,2,-1,-2];
147 S1tcv_mode=tcalc_mode_mapping(S1tcv+1)
148
149 % set Dwork(1) to temperature value
150 block.Dwork(1).Data=t0;
151 % set Dwork(2) to thermal comfort value
152 block.Dwork(2).Data=S1tcv_mode;
153
154 % Update Simulink model output ports
155 block.OutputPort(1).Data = t0;
156 block.OutputPort(2).Data = t0-2;
157 block.OutputPort(3).Data = t0-4;
158 block.OutputPort(4).Data = S1tcv_mode;
```

```
159 %endfunction: Outputs(block)
160
161 function Update(block)
162 %% Update Dwork
163
164 %endfunction: Update(block)
165
166 function Terminate(block)
167
168 %endfunction: Terminate(block)
```

Listing E.5 hil_readdata.m

E.7 hil_soc.m

```
function soc(block)
%% SOC *HIL
%
%  Title: State of Charge (HIL)
%  Filename: soc.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  Code tagged to Simulink model block soc_model.
%  Operates in 2 Modes: [0] normal operations [1] demand event
%  Initially SOC assumed 0 and will start to charge. At high
%  threshold ESS declared available for use (FIT=1). FIT status revert
%  back to 0 when low threshold reached (on discharge).
%  If SOC available and tariff HIGH (level 3), power switch to ESS (PWR=1).
%  When tariff LOW (level 1 or 2) power switch to GRID (PWR=0).
%  On receipt of demand event signal, MODE=0. Priority sets
%  ESS to charge during 4-hour ramp time before demand event starts and
%  power switch to GRID (PWR=0). At demand event start power switch to ESS
%  (PWR=1), ESS begins to discharge. Maintain ESS power for duration of
%  demand event. At end of demand event revert back to normal operations
%  (MODE=0). Self-Discharge Rate (SDR) applies on discharge.
%
%  >HIL
%  1. Delete all references to path_2; building subsystem removed from
%     Simulink model.
%

setup(block);

%endfunction: soc(block)

function setup(block)
%% Setup Functional Port Properties

% Register number of ports
block.NumInputPorts  = 4;
block.NumOutputPorts = 2;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Override input port properties
block.InputPort(1).Dimensions        = 1;  % DIR
block.InputPort(1).DatatypeID        = 8;  % boolean
block.InputPort(1).Complexity        = 'Real';
block.InputPort(1).DirectFeedthrough = true;

% Override input port properties
block.InputPort(2).Dimensions        = 1;  % DATA
block.InputPort(2).DatatypeID        = 0;  % double
block.InputPort(2).Complexity        = 'Real';
block.InputPort(2).DirectFeedthrough = true;

% Override input port properties
block.InputPort(3).Dimensions        = 1;  % t_mode
block.InputPort(3).DatatypeID        = 0;  % double
block.InputPort(3).Complexity        = 'Real';
block.InputPort(3).DirectFeedthrough = true;

% Override input port properties
block.InputPort(4).Dimensions        = 1;  % MODE
block.InputPort(4).DatatypeID        = 0;  % double
block.InputPort(4).Complexity        = 'Real';
block.InputPort(4).DirectFeedthrough = true;

% Override output port properties
block.OutputPort(1).Dimensions       = 1;  % FIT
block.OutputPort(1).DatatypeID       = 0;  % double
block.OutputPort(1).Complexity       = 'Real';

% Override output port properties
block.OutputPort(2).Dimensions       = 1;  % PWR
block.OutputPort(2).DatatypeID       = 0;  % double
block.OutputPort(2).Complexity       = 'Real';

```

```
77  % Register parameters
78  block.NumDialogPrms = 0;
79
80  % Register sample times
81  block.SampleTimes = [30 0];
82
83  % Specify the block simStateCompliance to default
84  block.SimStateCompliance = 'DefaultSimState';
85
86  % Register nethods
87  block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
88  block.RegBlockMethod('Start', @Start);
89  block.RegBlockMethod('Outputs', @Outputs); % required
90  block.RegBlockMethod('Update', @Update);
91  block.RegBlockMethod('Terminate', @Terminate); % required
92  block.RegBlockMethod('SetInputPortSamplingMode',@SetInpPortFrameData);
93
94  %endfunction: setup(block)
95
96  function DoPostPropSetup(block)
97
98  % Initialise the Dwork vectors
99  block.NumDworks = 1;
100
101 % DWork(1) store value at input port 2 [DATA] = raw SOC
102 block.Dwork(1).Name             = 'D1';
103 block.Dwork(1).Dimensions       = 1;
104 block.Dwork(1).DatatypeID       = 0;      % double
105 block.Dwork(1).Complexity       = 'Real'; % real
106 block.Dwork(1).UsedAsDiscState = true;
107
108 %endfunction: DoPostPropSetup(block)
109
110 function Start(block)
111 %% Set Start Conditions
112
113 % Assign Dwork(1) to 0
114 block.Dwork(1).Data = 0;
115
116 %endfunction: Start(block)
117
118 function Outputs(block)
119 %% Outputs
120
121 % define model paths
122 path_1='optim_ctrl_model_sim/scheduler_subsystem/ess_subsystem/SOC_hold';
123 path_3='optim_ctrl_model_sim/scheduler_subsystem/ess_subsystem/CD';
124
125 % Determine MODE: [0]=normal, [1]=demand event (ramp plus duration)
126 if (block.InputPort(4).Data==0)
127     % Normal Operations
128     set_param(path_1,'Value','1')
129     if (block.InputPort(1).Data==1) % DIR increasing (charge)
130         if (block.InputPort(2).Data>0.8) % detect SOC > 0.8
131             block.OutputPort(1).Data = 1; % FIT=1
132             if (block.InputPort(3).Data==3) % detect high tariff
133                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
134
135                 set_param(path_3,'Value','0')
136                 set_param(path_1,'Value','1')
137             else
138                 block.OutputPort(2).Data = 0; % PWR=0 (GRID)
139
140                 set_param(path_3,'Value','1')
141                 set_param(path_1,'Value','1')
142             end
143         else
144             block.OutputPort(2).Data = 0; % PWR=0
145
146             set_param(path_3,'Value','1')
147             set_param(path_1,'Value','1')
148         end
149     else % DIR decreasing (discharge)
150         if (block.InputPort(2).Data<0.2) % detect SOC < 0.2
151             block.OutputPort(1).Data = 0; % FIT=0
152             block.OutputPort(2).Data = 0; % PWR =0
153
154             set_param(path_3,'Value','1')
155             set_param(path_1,'Value','1')
156             return
157         else
158             if (block.InputPort(3).Data==3) % detect high tariff
```

```
159                 block.OutputPort(2).Data = 1; % PWR=1
160
161                 set_param(path_1,'Value','1')
162             else
163                 block.OutputPort(2).Data = 0; % PWR=0
164
165                 set_param(path_1,'Value','0');
166             end
167         end
168     end
169 else
170     % Demand Event (active for ramp plus duration)
171     if (block.InputPort(1).Data==1) % DIR increasing (charge)
172         if (block.InputPort(2).Data>0.90) % detect SOC > 0.95
173             block.OutputPort(1).Data = 1; % FIT=1
174             if (block.InputPort(3).Data==3) % detect high tariff
175                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
176
177                 set_param(path_3,'Value','0')
178                 set_param(path_1,'Value','1')
179             else
180                 block.OutputPort(2).Data = 0; % PWR=0 (GRID)
181
182                 set_param(path_3,'Value','1')
183                 set_param(path_1,'Value','1')
184             end
185         else
186             if (block.InputPort(3).Data==3) % detect high tariff
187                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
188
189                 set_param(path_3,'Value','0')
190                 set_param(path_1,'Value','1')
191             else
192                 block.OutputPort(2).Data = 0; % PWR=0
193
194                 set_param(path_3,'Value','1')
195                 set_param(path_1,'Value','1')
196             end
197         end
198     else % DIR decreasing (discharge)
199         if (block.InputPort(2).Data<0.2) % detect SOC < 0.2
200             block.OutputPort(1).Data = 0; % FIT=0
201             block.OutputPort(2).Data = 0; % PWR =0
202
203             set_param(path_3,'Value','1')
204             set_param(path_1,'Value','1')
205             return
206         else
207             if (block.InputPort(3).Data==3) % detect high tariff
208                 block.OutputPort(2).Data = 1; % PWR=1 (ESS)
209
210                 set_param(path_1,'Value','1')
211             else
212                 block.OutputPort(2).Data = 0; % PWR=0 (GRID)
213
214                 set_param(path_1,'Value','1');
215                 set_param(path_3,'Value','1')
216             end
217         end
218     end
219 end
220
221 %endfunction: Outputs(block)
222
223 function Update(block)
224 %% Update Dwork
225
226 % Update Dwork(1) to InputPort(2) [Data] = raw SOC
227 block.Dwork(1).Data = block.InputPort(2).Data;
228
229 %endfunction: Update(block)
230
231 function SetInpPortFrameData(block, idx, fd)
232
233 % Set the sampling of the input ports
234 block.InputPort(idx).SamplingMode=fd;
235 for i=1:block.NumOutputPorts
236     block.OutputPort(i).SamplingMode=fd;
237 end
238
239 %endfunction: SetInpPortFrameData(block)
240
```

```
241  function Terminate(block)
242
243  %endfunction: Terminate(block)
```

Listing E.6 hil_soc.m

E.8 hil_te2u.m

```
function u =te2u(Te)
%% MATLAB Function Description *HIL
%
%  Title: Convert Temperature Error to Control Action (HIL)
%  Filename: prepare_comfort_values.m
%  Prepared by: Sean Williams
%  Date: 6 Nov 2019
%
%  MATLAB function sets temperature setpoint (control action) depending on
%  measured temperature. System limited to operate in temperature range
%  15.5degC (minimum) to 20.5degC (maximum).
%

%% Change History
%
%   1. [06-11-2019] Initial
%   2. [21-01-2020] Set upper temperature range to 20.5 irrespective of
%   measured temperature (last line in ifelse block set to 23.00)
%   3.
%

%% Convert Temperature Error to Control Action (Te2u)
% map Te to control action voltage (0-10Vdc)
Tu=0;
if (Te ≥ 0) && (Te < 0.3)
    Tu=0;
elseif (Te ≥ 0.3) && (Te < 0.5)
    Tu=1;
elseif (Te ≥ 0.5) && (Te < 1.0)
    Tu=2;
elseif (Te ≥ 1.0) && (Te < 1.5)
    Tu=3;
elseif (Te ≥ 1.5) && (Te < 2.0)
    Tu=4;
elseif (Te ≥ 2.0) && (Te < 2.5)
    Tu=5;
elseif (Te ≥ 2.5) && (Te < 3.0)
    Tu=6;
elseif (Te ≥ 3.0) && (Te < 3.5)
    Tu=7;
elseif (Te ≥ 3.5) && (Te < 4.0)
    Tu=8;
elseif (Te ≥ 4.0) && (Te < 4.5)
    Tu=9;
elseif (Te ≥ 4.5)
    Tu=10;
end

u=Tu;

end
% end prepare_comfort_values
```

Listing E.7 hil_te2u.m

E.9 hil_write_serialdata.m

```
function write_serialdata(data_ctrl_action)
%% MATLAB Function Description *HIL
%
%  Title: Write Serial Data (HIL)
%  Filename: write_serialdata.m
%  Prepared by: Sean Williams
%  Date: 15 Mar 2020
%
%  MATLAB function writes control action data to serial port.
%  COM port connects to Industruino using USB.
%

%% Change History
%
%   1. [15-03-2020] Initial
%   2.

%% Write Serial Data
% to enable data transfer data_ctrl_action parameter is multipled by 100
% and rounded before TX. Industruino RX divide by 100 to restore value
data_ctrl_action=round(data_ctrl_action*100,0);

%  set port parameters
s=serial('COM16'); % USB to Industruino
set(s,'BaudRate',9600);
set(s,'DataBits',8);
set(s,'StopBits',1);
set(s,'Parity','none');
set(s,'Timeout',8);
set(s,'Terminator','CR');

% open port
fopen(s)

% write data_ctrl_action
fprintf(s,'%d\n', data_ctrl_action)

% close port
fclose(s)

end
% end write_serialdata
```

Listing E.8 hil_write_serialdata.m

E.10 hil_writedata.m

```
function writedata(block)
%% WRITEDATA *HIL
%
%  Title: Write Data (HIL)
%  Filename: writedata.m
%  Prepared by: Sean Williams
%  Date: 9 Mar 2020
%
%  Write control action value to serial port
%  Sample rate is 5min (300s)
%

setup(block);

%endfunction: writedata(block)

function setup(block)
%% Setup Functional Port Properties

% Register number of ports
block.NumInputPorts  = 1;
block.NumOutputPorts = 0;

% Setup port properties to be inherited or dynamic

block.SetPreCompOutPortInfoToDynamic;

% Override output port properties
block.InputPort(1).Dimensions  = 1; % ctrl_action
block.InputPort(1).DatatypeID  = 0; % double
block.InputPort(1).Complexity  = 'Real';
block.InputPort(1).SamplingMode='Sample';

% Register parameters
block.NumDialogPrms = 0;

% Register sample times
block.SampleTimes = [300  0];

% Specify the block simStateCompliance to default
block.SimStateCompliance = 'DefaultSimState';

% Register nethods
block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('Start', @Start);
block.RegBlockMethod('Outputs', @Outputs); % Required
block.RegBlockMethod('Update', @Update);
block.RegBlockMethod('Terminate', @Terminate); % Required
block.RegBlockMethod('SetInputPortSamplingMode',@SetInpPortFrameData);

%endfunction: setup(block)

function DoPostPropSetup(block)

% Initialise the Dwork vectors
block.NumDworks = 1;

% Dwork(1) stores the value of the last control action reading
block.Dwork(1).Name            = 'lastctrl_action';
block.Dwork(1).Dimensions      = 1;
block.Dwork(1).DatatypeID      = 0;      % double
block.Dwork(1).Complexity      = 'Real'; % real
block.Dwork(1).UsedAsDiscState = true;

%endfunction: DoPostPropSetup(block)

function InitializeConditions(block)
block.Dwork(1).Data=0;

%endfunction: InitializeConditions(block)

function Start(block)
%% Set Start Conditions

% Assign Dwork(1) to status of last temp
```

```
77  block.Dwork(1).Data=0;
78 %endfunction: Start(block)
79
80  function Outputs(block)
81 %% Outputs
82 % hold previous value of data_ctrl_action
83  lastctrl_action=block.Dwork(1).Data;
84
85 % set data_ctrl_action value to InputPort(1)
86  data_ctrl_action=block.InputPort(1).Data;
87
88 % write data_ctrl_action (ctrl_action) to serial out
89  write_serialdata(data_ctrl_action)
90
91 % set Dwork(1) to current value of data_ctrl_action
92  block.Dwork(1).Data=data_ctrl_action;
93
94 %endfunction: Outputs(block)
95
96  function Update(block)
97 %% Update Dwork
98
99 %endfunction: Update(block)
100
101  function Terminate(block)
102
103 %endfunction: Terminate(block)
```

Listing E.9 hil_writedata.m